A DATA-QUALITY DRIVEN FRAMEWORK FOR DATA DISSEMINATION IN WIRELESS SENSOR NETWORKS

BY

WEI-PENG CHEN

B.S., National Taiwan University, 1994
M.S., National Taiwan University, 1996
M.S., The Ohio State University, 2001

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2004

Urbana, Illinois

Abstract

Distinct from wireless ad hoc networks, wireless sensor networks are data-centric, application-oriented, collaborative, and resource-constrained in nature. Especially, the limited resources on energy, bandwidth and computation capability radically change the considerations of system design.

In this thesis we propose a comprehensive data-centric information processing and dissemination framework for sensor networks. We consider two main research issues in this thesis: (**T1**) light-weight coordination mechanisms among sensors to collect valuable information from the environment and (**T2**) efficient dissemination methods to deliver the information of the best quality from sensors to subscribers. For the first issue, we propose a self-organized, dynamic clustering approach for target tracking. Coordination between sensors is triggered by the events of interests and a cluster consisting of a leader and several sensors is formed dynamically. With the use of a probabilistic leader volunteering procedure and a sensor replying method based on the quality of data, the proposed dynamic clustering algorithm effectively eliminates contention among sensors and renders more accurate estimates of target locations. For the second issue, we consider the problems of routing and data transport and formulate as a utility-based optimization problem, with the objective of maximizing the amount of information (utility) collected at sinks (subscribers), subject to flow conservation, channel bandwidth, and energy constraints. Both the centralized and distributed approaches are devised to solve the optimization problem. To validate the design and to empirically study the performance of the proposed works, we implement a subset of acoustic tracking and utility-based data transport components on the Motes testbed, and demonstrate the value of incorporating information quality in data transport.

To my parents.

Acknowledgments

This thesis would not have been possible without the help of my advisors Prof. Jennifer C. Hou and Prof. Lui Sha. Prof. Hou has led me into the field of networking, guided me through various research processes, and supported me financially during my graduate study. Prof. Sha has taught me the correct method of conducting research and contributed invaluable suggestions in my thesis. I also would like to thank Prof. P.R. Kumar and Prof. Klara Nahrstedt for their insightful comments while serving on my thesis committee, and am grateful to Prof. Marco Caccamo for collaborating on sensor networks research. I am grateful to my graduate colleagues, Hung-Ying Tyan, Yung-Ching Hsiao, Ye Ge, Yuan Gao, Rong Zheng, Hwangnam Kim, Lu-chuan Kung, Guanghui He, Honghai Zhang, Ning Li, Ajay S. Triumala, Qixin Wang, and Ahmed Sobeih, for their intellectual research discussion and friendship. Last but certainly not least, I would most like to express my appreciation to my parents, my sisters, and my girl friend, Tang-chih, for their endless love, encouragement, and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recent technological advances have led to the emergence of small, low-power devices that integrate sensors and actuators with limited on-board processing and wireless communication capabilities. Pervasive networks of such sensors and actuators open new vistas for constructing complex monitoring and control systems, ranging from habitat monitoring [56], target tracking system [94], home automation [36], to ubiquitous computing [76, 32]. With the intrinsic constraints in size and cost, sensor networks possess certain distinct characteristics which warrant their treatment as a special class of ad hoc networks:

1. *Data-centric*: Sensor networks are largely data-centric, with the objective of delivering collected data, in a timely fashion, to destinations that require such data. Data that contains information of different qualities represents different values to destinations. As a result, the overall system objective is no longer to maximize the raw data throughput. Instead, maximizing the amount of useful information carried to destinations is an important criterion.

2. *Application-oriented*: While traditional wired and wireless networks are expected to cater to a variety of applications, sensor networks are usually deployed to perform specific tasks. The specific algorithms/protocols and performance metrics used in sensor networks thus depend on the characteristics and requirements of applications. For instance, for mission-critical applications, it is very important to ensure the end-to-end latency be kept below certain threshold.

3. *Collaborative*: Because of the application-oriented nature of sensor networks, how nodes collaborates with each other to realize the global system objective outweigh the objective of

achieving fairness of individual connections. This is in sharp contrast to conventional wired and wireless networks in which provisioning of fairness to users is an important design criterion.

4. *Resource-constrained*: Compared to traditional wired or wireless networks, sensor networks impose more constraints on resource usage on energy, bandwidth, and computation. First, as most of the low-power devices in sensor networks have limited battery life and replacing batteries on tens of thousands of these devices is infeasible, any protocol/algorithm deployed in sensor networks has to be energy aware. Second, due to the limited available bandwidth and the hardware constraints of radio circuitry, sensor networks cannot provide high throughput in transporting data. Any data must be digested before being transmitted to other nodes. In-network processing and data fusion are the keys to reduce bandwidth usage. Finally, workload should be properly distributed to each sensor. Network heterogeneity is a necessity in some situations in which the tasks with heavy computation requirement can be executed at high-capability sensors.

1.1 Overview of the Thesis

In this thesis, we propose a comprehensive, data-centric information processing and dissemination framework for sensor networks. The framework is composed of two phases: (i) data fusion takes place at a group of geographically proximate sensors and (ii) data dissemination based on data quality to transport as much information as possible from different aggregators to users. The two main research issues investigated in this thesis are: (**T1**) light-weight coordination mechanisms among sensors to collect useful information from environment and (**T2**) efficient dissemination methods to deliver information of the best quality from sensors to subscribers. For the first issue, we propose a self-organized, dynamic clustering mechanism [13] for target tracking. Coordination between sensors is triggered by events of interests and a cluster consisting of a leader and several sensors is formed dynamically. With the use of a probabilistic leader volunteering procedure and a sensor replying method based on the quality of data, the proposed dynamic clustering algorithm effectively eliminates contention among sensors and renders more accurate estimates of target locations. For the second issue, we consider the problems of routing and data transport by formulating

a utility-based optimization problem, with the objective of maximizing the amount of information (utility) collected at sinks (subscribers), subject to flow conservation, channel bandwidth, and energy constraints. Both the centralized [14] and distributed [15] approaches are devised to solve the optimization problem.

In what follows, we first give in Chapter 2 a comprehensive overview of existing work that has been performed in data aggregation and dissemination and then elaborate on, for each issue, the technical motivation, our proposed approach, and key results.

1.2 Dynamic Clustering for Acoustic Target Tracking System in Wireless Sensor Networks

One of their most important applications is target tracking, with the targets to be tracked ranging from security attacks in the forms of chemical, biological, or radiological weapons, to moving objects in civil surveillance, and to changes in light, temperature, pressure, acoustics in environmental monitoring. The type of signals to be sensed is determined based on the types of objects to be tracked. In spite of the different targets to be tracked and the various signals to be sensed, tracking applications share several common characteristics: first, the tracking system should report the location of the target to subscribers (usually remote controllers) accurately and in a timely manner. Second, because the data collected by sensors may be redundant, correlated, and/or inconsistent, it is desirable to have sensors collaborate on processing the data and transporting a concise digest to subscribers. This reduces not only the number of packets to be transported, but also the probability of collision and interference in the shared media. *Localized and collaborative* data processing aids in reducing the power consumed in communication activities and hence prolonging the lifetime of sensor networks.

To facilitate collaborative data processing in target tracking-centric sensor networks, the cluster architecture is usually used in which sensors are organized into clusters, with each cluster consisting of a cluster head (CH) and several neighboring sensors (members). In the conventional cluster architecture, clusters are formed statically at the time of network deployment. The attributes of each cluster, such as the size of a cluster, the area it covers, and the members it possesses, are static. In spite of its simplicity, the static cluster architecture is not robust from the perspective of fault tolerance. To deal with the several problems that arise in static clustering, we devise and

evaluate in Chapter 3 a fully decentralized, light-weight, dynamic clustering algorithm for target tracking. Formation of a cluster is triggered by certain events of interest (e.g., detection of an approaching target with acoustic sounds) and the dynamic cluster intends to follow the movement of the target. Instead of assuming the same role for all the sensors, we envision a hierarchical sensor network that is composed of (a) a static backbone of sparsely placed high-capability sensors which will assume the role of a cluster head (CH) upon triggered by certain signal events; and (b) moderately to densely populated low-end sensors whose function is to provide sensor information to CHs upon request. A cluster is formed and a CH becomes active, when the acoustic signal strength detected by the CH exceeds a pre-determined threshold. The active CH then broadcasts an information solicitation packet, asking sensors in its vicinity to join the cluster and provide their sensing information.

We address and devise solution approaches (with the use of Voronoi diagram) to realize dynamic clustering: (I1) how CHs cooperate with one another to ensure that only one CH (preferably the CH that is closest to the target) is active with high probability ; (I2) when the active CH solicits for sensor information, instead of having all the sensors in its vicinity reply, only a sufficient number of sensors respond with non-redundant, essential information to determine the target location; and and (I3) both the packets that sensors send to their CHs and packets that CHs report to subscribers do not incur significant collision.

Through both the probabilistic analysis and *ns-2* simulation, we evaluate and demonstrate the effectiveness of the proposed approaches. In particular, we show via a simplified model (which captures all the essential properties) that the probability that packets collide with one another is very small under the proposed approaches. We also show via simulation that with the use of Voronoi diagram, the CH that is usually closest to the target is (implicitly) selected as the leader, and that the proposed dynamic clustering algorithm effectively eliminates contention among sensors and renders better and more accurate estimates on the target location as a result of better quality data collected and less collision incurred. As compared to the performance of the static cluster architecture, the proposed approaches reduce the estimation error and latency by 37% and 16%, respectively. By combining a more sophisticated localization method, the performance in the estimation error can be further improved to be 71%.

1.3 Energy-Aware Data-Centric Utility-Based Approach

After the sensors extract the valuable information, the next challenge is to disseminate the information to the remote subscribers through multi-hops wireless links. As a result of the unique characteristics of sensor networks mentioned above, conventional routing and flow control protocols that focus on maximizing raw data throughput and achieving fairness are no longer well suited for sensor networks. Instead, the notion of *directed diffusion* is proposed in [35]. However, it does not take into consideration of resource utilization, especially under the case that numerous queries are from users simultaneously. A mechanism properly controlling the usage of scarce resource to deliver the most useful information to the sink is required. On the other hand, existing data transport protocols [79, 80, 66, 28, 83] for wireless sensor networks consider the issues of reliability and congestion control but do not figure in one of the most important characteristics – *data-centric*. We believe data-centric, utility based approaches that differentiate treatments of packets with respect to their utility values and at the same time, take into account of both bandwidth usage and energy consumption are more adequate. Such mechanisms can solve jointly the problems of maximizing utility and mitigating congestion.

In this thesis, we formulate the problem of data transport in sensor networks as an optimization problem whose objective function is to maximize the amount of information (utility) collected at sinks (subscribers), subject to the flow, energy and channel bandwidth constraints. In Chapter 4, we consider a non-convex programming problem formulation and show that it is general enough to encompass a wide variety of applications in sensor networks, each with a different objective function and subject to different constraints. Specifically, we consider six design dimensions and adapt the generic formulation to meet the various needs of different applications. Also, based on a Markov model extended from [6], we derive the link delay and the node capacity in both the single and multi-hop environments, and figure them in the problem formulation. The simulation results show that the proposed energy-aware flow control solution can achieve high utility and low delay without congesting the network.

The problem formulation in Chapter 4 is a non-convex programming problem and a centralized approach is used to solve the optimization problem. As the centralized approach cannot quickly adapts to dynamic network changes, we devise in Chapter 5 a distributed solution by transforming

the optimization problem to a convex programming problem, and explore in three directions. First, we devise a simple node capacity estimation method to on-line measure the node capacity (which changes with the traffic load and nodal distribution and is required in the optimization problem). Second, we linearize the energy constraint by properly setting the value of the system lifetime in advance and controlling the data rate of a node (and hence its total energy consumption rate) so as to sustain its battery lifetime longer than the specified lifetime. Finally, we incorporate the optimization results into routing so as to provide sensors with opportunities to select better routes. The simulation results show that the utility-based approach balances between system utility and system lifetime.

1.4 Empirical Study of the Proposed Approaches

To validate the design and to empirically study the performance of the proposed work, we present in Chapter 6 an implementation of a subset of our target tracking and utility-based data transport work on the testbed with the tiny wireless sensors, Motes [18]. The major purpose of the implementation is to lay a generic software architecture along with its well defined APIs for the sensor network community to realize data-centric information dissemination mechanisms. Both the delay-based and energy-based localization methods are also implemented as utility function; they render localization errors within 30 inches. To demonstrate the use of the software architecture we have implemented a simplified version of the dynamic clustering protocol given in Chapter 3 and a subset of the distributed utility-based approach given in Chapter 5. A sample application are provided to demonstrate the value of the utility based data transport. The proposed software architecture can be easily adapted to various characteristics and requirements of different applications.

The rest of the proposal is organized as follows. We summarize all the related work in Chapter 2. The dynamic clustering for target tracking system is presented in Chapter 3. Following that we delve into a centralized and a distributed utility-based approach in Chapter 4–5, respectively. The empirical study on Berkeley Motes is given in Chapter 6. Finally we conclude the thesis and list the future research directions in Chapter 7.

Chapter 2

Related Work

Data gathering is one of the most important tasks in sensor networks. In most data gathering applications, sensors extract useful information from the environment, and either respond to queries made by users or take the active role to disseminate the information to one or more sinks. The information is then exploited by subscribers/users for environment monitoring, target tracking, and/or decision making. In some sense, a sensor network can be envisioned as a distributed database that provides a layer of query processing for users. Research issues of how information can be effectively gathered, aggregated, and disseminated to users and how queries made by users can be effectively directed to sensors that have the corresponding information have been addressed extensively. In this chapter, we give an overview of research activities on data gathering and fusion along four research thrusts: (1) query processing in sensor database systems, (2) data gathering and dissemination mechanisms, (3) data fusion mechanisms, and (4) optimization in data gathering.

2.1 Overview

Recent technological advances have led to the emergence of small, low-power devices that integrate sensors and actuators with limited on-board processing and wireless communication capabilities. Pervasive networks of such sensors and actuators open new vistas for constructing complex monitoring and control systems, ranging from habitat monitoring [56], target tracking [94], home automation [36], ubiquitous sensing for smart environments [22], construction safety monitoring, and inventory tracking. In most sensor network applications, sensors extract useful information from the environment, and either respond to queries made by users or take the active role to disseminate the information to one or more sinks. The information is then exploited by subscribers/users for

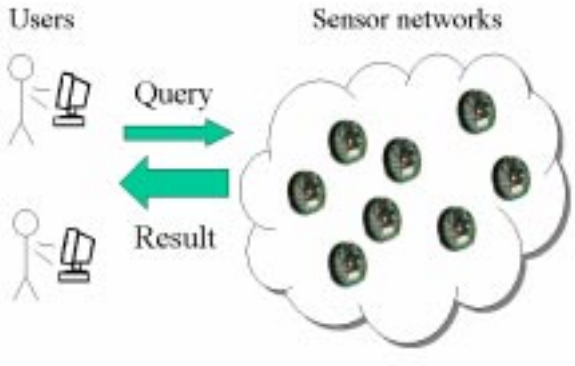


Figure 2.1: Query/Result relationship between users and sensor networks.

their decision making. In other words, one can envision sensor networks as a distributed database for users to query the physical world [7]. Fig. 2.1 depicts the simplified relationship between users and sensor networks.

The process of data gathering in sensor networks, nevertheless, is significantly different from conventional warehousing database systems where data are extracted from sensors and stored in a centralized server that is responsible for query processing. Aside from the fact that sensor networks operate in a distributed fashion, they encompass several distinct characteristics and hence pose more challenges [23, 24]: (1) the convention that sensors are usually deployed with a high nodal density poses the scalability problem; (2) the fact that these sensors are usually left unattended once deployed makes autonomous operations necessary; (3) the fact that the computing and communication environment is unreliable due to the irregular terrain, environment dynamics, energy depletion, and potential hardware defects requires that the design be robust; and (4) the resource constraints in energy, bandwidth, storage and computation capability requires that resources be efficiently used. In general, the design criteria for data gathering applications in sensor network are: (1) scalability, (2) autonomy, (3) robustness, and (4) energy-efficiency. In addition, there are several features that should be included in the design and implementation of data gathering applications:

(1) Devising localized algorithms: In a localized algorithm, each node operates on the information locally collected. This feature enables the algorithm to incur much less communication exchanges (and hence save power) in the case of environmental stimuli and topology changes

(as a result of power depletion) and hence to adapt more easily to these changes. Well- controlled communication overheads (and hence the saving in power) also ensure scalability of localized algorithms.

(2) Aggregating data in the process of routing [42]: Redundancy exists in sensor data in both the temporal and spatial domains. That is, readings collected by a single sensor at different times or readings collected among neighboring sensors may be highly correlated and contain redundant information. Instead of transmitting all the highly correlated information to subscribers, it may be more effective for some intermediate sensor node(s) to digest the information received and come up with a concise digest, so as to reduce the amount of raw data to be transmitted (and hence the power incurred, and bandwidth consumed, in transmission). This technique is termed as data fusion (a. k. a. data aggregation). Data fusion can also be combined with routing. A sensor node on the route first aggregates data from its previous hop and its own data, and forwards the digested information to the next hop towards the destination. Compared with traditional address-centric routing that finds the shortest paths between pairs of end-nodes, data-fusion-centric routing aims to locate routes that lead to the largest degree of data aggregation.

(3) Being adaptive to topology changes: Due to environmental dynamics (such as channel fading due to weather effects) and nodes failures (due to power depletion and hardware failure), the network topology may change from time to time. In addition, the traffic source and destination as well as the traffic amount may vary. Adaptation to these changes is the key to make the system autonomous and efficient.

(4) Increasing node/route redundancy: In (2) we state that it is desirable to remove the data redundancy in the time and spatial domains. On the other hand, deploying a sensor network with a high nodal density so as to increase node redundancy (and hence route redundancy) is likely to make the system more resilient and robust to all the aforementioned environment dynamics. Increasing node redundancy also extends the network lifetime if the sensing and monitoring task can be rotated among several disjoint sets of nodes, each of which takes turns to become active in monitoring the entire region.

In this chapter, we give a survey of research activities in the areas of data gathering, dissemination, and fusion. The survey is conducted along four research thrusts: (1) query processing in sensor database systems, (2) data gathering and dissemination mechanisms, (3) data fusion mechanisms, and (4) optimization in data gathering. The categorization is made roughly based on the major focus of algorithms, although some algorithms consider both data dissemination and fusion jointly. The rest of this chapter is organized as follows. In Section 2.2 we introduce sensor database systems and how queries are processed in such systems. In Section 2.3, we present an overview of data gathering and dissemination mechanisms and two predominant factors that determine the system architecture. A taxonomy of data gathering mechanisms based on storage locations, directions of diffusion, and structures of dissemination is presented in Sections 2.3.1- 2.3.3. Following that, we give an overview of data fusion mechanisms in Section 2.4. A classification of data fusion approaches based on functions of data fusion, system architectures, and tradeoffs in the system design is then treated in Sections 2.4.1- 2.4.3. Finally, we present several data gathering approaches that maximize the amount of information extracted in Section 2.6.

2.2 Sensor Database

Sensor networks provide a new computing platform for users to readily access the data in the physical world [7]. They can be viewed as a large distributed database system. Consider an environment monitoring and alert system that is similar to the ALERT system (http://www.alertsystems.org). Several types of sensors including rainfall sensors, water level sensors, weather sensors, and chemical sensors are used to record the precipitation and water level regularly, to report the current weather condition, and to issue flood warnings or chemical pollution. In such a monitoring application, there are five types of queries that users typically make [7, 55, 47]:

(1) Historical queries: These are aggregate queries over the historical data stored at a database system, e.g., "What is the average level of rainfall of Champaign County in May 2000?"

(2) Snapshot queries: These queries are concerned with the information gathered from the network at a specified (current or future) time point, e.g., "Retrieve the current readings of temperature sensors in Champaign County."

(3) Long-running queries: These queries last for a period of time, e.g. "Retrieve every 30 min-

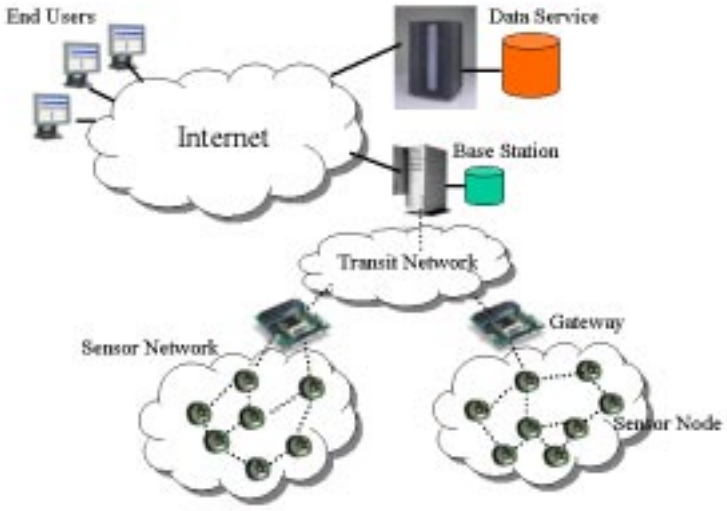


Figure 2.2: The complete architecture of a sensor database system [56]

utes the highest temperature sensor reading in Champaign County from 6:00pm to 10:00pm tonight."

(4) Event-triggered queries [55]: These queries pre-specify the thresholds or conditions which trigger a query, e.g., "If the water level exceeds 10 meters in Champaign County, query the rain fall sensors about the amount of precipitation during the past hour. If the amount of precipitation exceeds 100 mm, send an emergency message to the base station to issue a flood warning."

(5) Multi-dimensional range queries [47]: These queries involve more than one attribute of sensor data and specify the desired search range as well, e.g., "In Champaign County, list the positions of all sensors that detect water level between 5 to 8 meters and have temperatures between 50 and 60 °F."

A complete hierarchical architecture (4-tier) of sensor database systems for a monitoring application answering the above five types of queries is depicted in Fig. 2.2. The lowest level is a group of sensor nodes that performs sensing, computing, and in-network processing in a field. The data collected within the sensor networks are first propagated to its gateway node (second level). Next the gateway node relays the data through a transit network to a remote base station (third level). Finally, the base station connects to a database replica across the Internet. Among the four

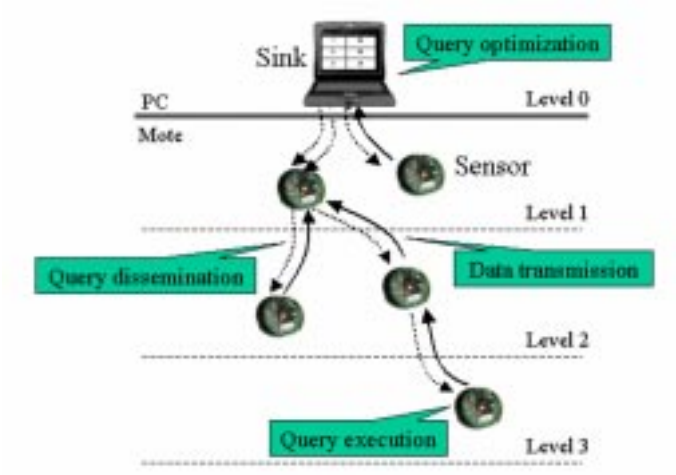


Figure 2.3: Procedures for query and data extraction in TinyDB [26, 55].

tiers of the system, the resource within the sensor networks is the most constrained. In most of the applications the sensor network is composed of sensors and a gateway node (sink) as shown in Fig. 2.3 although the number of sinks or sources might vary from application to application.

2.2.1 Example Sensor Database System

The main purpose of a sensor database system is to facilitate the data collection process. Users specify their interests via simple, declarative SQL-like queries. Upon receipt of a request, the sensor database system efficiently collects and processes data within the sensor network, and disseminates the result to users [26]. A layer of query processing between the application layer and the network layer provides an interface for users to interact with sensor networks. The layer should also be responsible for managing the resource (especially the power) . Two of the most representative sensor database systems are TinyDB [55, 101] and Cougar [88, 26]. The former evolves from Tiny AGgregation (TAG), is developed by University of California at Berkeley, and is built on top of TinyOS operating system [102] and the platform of Motes, while the latter by Cornell University. Both the TinyDB and Cougar architectures consist of a single base station (sink) and multiple sensors. The sink and sensors are connected in a routing tree shown in Fig. 2.3. A sensor chooses its parent node that is one hop closer to the root (sink). The sink accepts queries from users outside the sensor network. Query processing can be divided into four steps: query optimization, query dissemination, query execution, and data dissemination.

Both TinyDB and Cougar provide a declarative SQL-like query interface for users to specify the

data to be extracted. Similar to SQL, the acquisitional query language used in TinyDB, TinySQL, consists of a **SELECT** – **FROM** – **WHERE** clause supporting selection, join, projection and aggregation. The data within sensor networks can be virtually considered as a table, each column of which corresponds to an attribute and each raw of which corresponds to a sample measured at a specific location and time. An example in TinySQL is like:

SELECT region id, **AVG**(water level), **AVG**(precipitation)
FROM water level sensor (W), precipitation sensor (P)
WHERE W.location **IN** Champaign County **AND** P.location IN Champaign County
GROUP BY region
Having AVG(W.water level) > 10 meters
EPOCH DURATION 10 minutes
TRIGGER ACTION report an emergency warning

The above query continually monitors the water level in all regions in Champaign County every 10 minutes. If the average water level of sensors in a region exceeds 10 meters, the system generates a flooding warning and sends the region id, the value of the average water level, and precipitation to the sink. The query language in sensor database mainly differs from SQL in that its queries are continuous and periodic [26]. Upon reception of a query, the sink performs query optimization to reduce the energy incurred in the pending query process. Two query optimization techniques are commonly used in TinyDB: ordering of sampling operations and query aggregation. First, since the energy incurred in retrieving readings form different types of sensors is different, the sampling operations should be reduced for sensors consuming higher energy. For instance, the energy consumed for sampling a magnetic reading is much higher than that for a light reading. The sampling energy can be saved if a proper ordering of sampling operations can be arranged in the evaluation of **HAVING** clause. For another example, the query "**HAVING** light > 200 and mag > 100" consumes less energy than the query "**HAVING** mag > 100 and light > 200" because in the former case the sampling operation for magnetic readings can be skipped if the condition on the light reading fails. Second, by combining multiple queries for the same event into a single query, only one query is sent. After a query is optimized at the sink, it is broadcast by the sink and disseminated

to the sensor network. When a sensor receives a query, it has to decide whether to process the query locally and/or rebroadcasts it to its children. A sensor only needs to forward the query to its children nodes that may have the matched result to save energy on disseminating and processing queries. To this end, a sensor has to maintain information of its children's attribute values. In TinyDB, a semantic routing tree (SRT) containing the range of the attributes of its children is constructed at each sensor. The attributes can be static information (e.g., location) or dynamic information (e.g., light readings). For attributes that are highly correlated among neighbors in the tree, SRT can reduce the amount of disseminated queries. One distinct characteristic of query execution in TinyDB is that sensors sleep during every epoch but are synchronized to wake up, receive, transmit, and process the data in the same time period.

2.3 Data Gathering and Dissemination Mechanisms

The wide variety of requirements and objectives for different applications in sensor networks imposes various design criteria and in turns leads to different solutions. Two major factors that determine the system architecture and design methodology are:

(1) The number of sources and sinks within the sensor network: Sensor network applications can be classified into three categories: one-sink-multiple-sources, one-source-multiple-sinks, and multiple-sinks-multiple-sources. An environment monitoring application shown in Fig. 2.2 falls in the one-sink-multiple-sources category since the interaction between the sensor network and the subscribers is usually through a single gateway (sink) node. On the other hand, a traffic reporting system where a single accident (source) is disseminated to many drivers (sinks) in the vicinity of the source falls in the one-source-multiple-sinks category. Applications can be further classified based on their storage location, direction of diffusion, and structure devices.

(2) The tradeoffs between energy, bandwidth, latency and information accuracy: An approach cannot usually optimize its performance in all aspects (e.g., energy usage, bandwidth usage, latency, and estimation accuracy). Instead, based on its special requirements, an application usually trades less important criteria for optimizing the performance with respect to the most important attribute. For instance, for mission-critical applications, the end-to-end latency is

perhaps the most important attribute and needs to be kept below certain threshold, even at the expense of additional energy consumption.

In what follows, we categorize data gathering and dissemination mechanisms based on the following three factors: (1) storage location, (2) direction of diffusion, and (3) structure of devices.

2.3.1 Classification of Data Gathering Mechanisms Based on the Storage Location

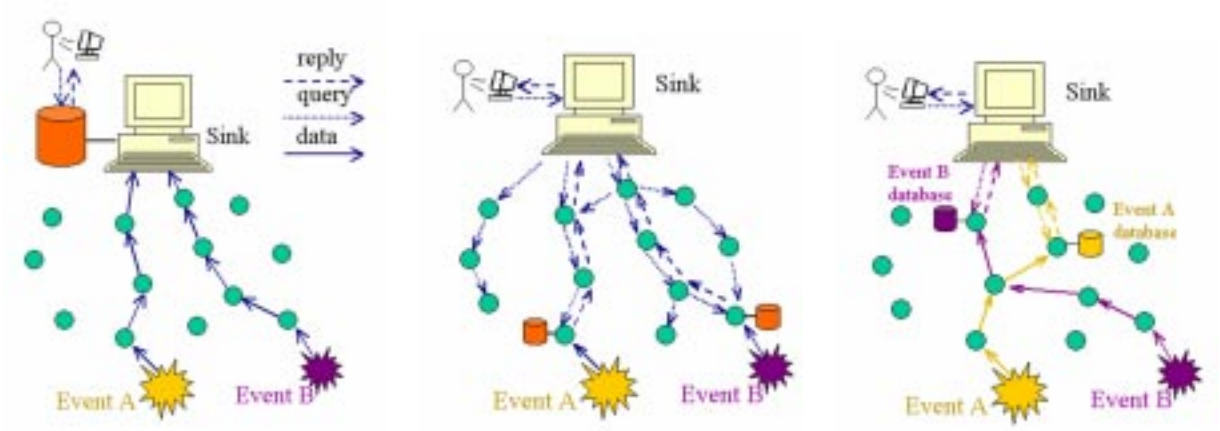
In order to process historical queries, data collected at different sensors have to be properly stored in a database system for future query processing. Fig. 2.4 shows three scenarios of placing storage at different locations [64]:

1. External Storage (ES): All the data collected at sensors are relayed to the sink and stored at its storage for further processing. For a sensornet with n sensor nodes, the cost of transmitting data to the external storage is $O(\sqrt{n})$. There is no cost for external queries, while the cost of a query within the networks incurs a cost of $O(\sqrt{n})$.

2. Local Storage (LS): Data is stored at each sensor's local storage and thus no communication costs for data is incurred. However, each sensor needs to process all queries and a query is flooded to all sensors. The cost of flooding a query is $O(n)$.

3. Data-Centric Storage (DCS): DCS stores the data at a sensor (or a location) within the sensor network based on the content of the data. A DCS system includes two components: first the sensor maps an event it detects to a label via a consensus hash function and then routes the data to a node according the label. The label can be a location and the sensor can route the data via geographic routing. We will introduce two of the representative approaches relying on geographic information, GHT [64] and DIM [47] below. Both data and query communication costs are $O(\sqrt{n})$.

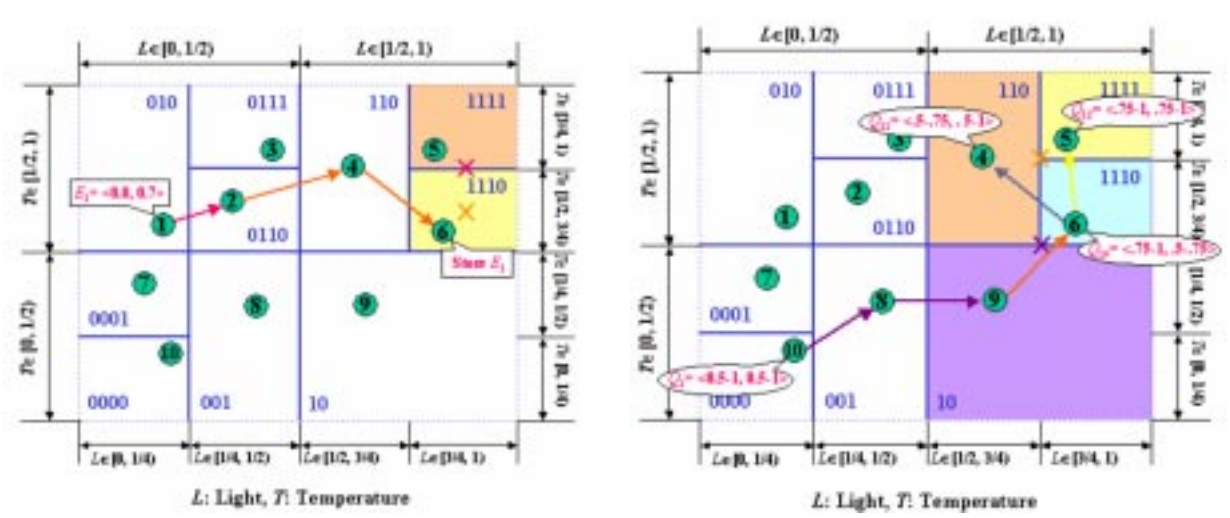
Database with Geographic Information: As mentioned above, one of the common hash functions in sensor database systems is to map the data to a location and then deliver via geographic routing, the data to the sensor node that is closest to the mapped location for storage. If all of the sensors have the same hash function, a query with a specific content can be converted to a



(a) External Storage (b) Local Storage (c) Data-Centric Storage

Figure 2.4: Three types of storage scenarios [64]

location where the data is stored for future retrieval. Geographic Hash Table (GHT) [64] and Distributed Index for Multi-dimensional data (DIM [47] are two of the representative databases with geographic information. Both of them adopt GPSR (Greedy Perimeter Stateless Routing) [39] as the underlying routing protocol, but their hash functions are slightly different. In GHT, the input to the hash function is a reading of a single attribute or one type of event, and the hash result is a point in the 2-D space. If no sensor node is located at the precise coordinates of the hash result, the data is stored at the node closest to the location of the hash result. With the use of the perimeter mode of GPSR, the data packet traverses the entire perimeter enclosing the location of the hash result and thus the closest location can be identified. DIM, on the other hand, is designed especially for multi-dimensional range queries. DIM maps a vector of readings with multiple attributes to a 2-D geographic zone. Two assumptions are made in DIM: first sensors are aware of their own locations and field boundary, and second all the sensors are static. The entire field is divided recursively into zones as shown in Fig. 2.5. The sequences of divisions are vertical, horizontal, and so on. Each zone is encoded with a unique code based on the following rule: For a vertical division (the i^{th} division where i is an odd number), the i^{th} bit code of the zone is encoded as "1" if it is in the right region; otherwise the i^{th} bit is encoded as "0". Similarly, the even bit of the codeword is determined by whether the zone is above ("1") or below ("0") the divided line. For instance, the codeword of the region in which node 6 resides in Fig.2.5 is "1110". Due to the fact that sensors may not be uniformly deployed in an area, every zone defined above may not contain



(a) Inserting an event

(b) Issuing a multi-dimension range query.

Figure 2.5: Operations of insertion and query in DIM [47].

a sensor. In other words, a sensor needs to determine its owned zone where no other sensors reside. This can be easily achieved when a node is aware of its neighbors' locations.

The encoding rule for mapping an event A with m normalized attributes $(A_1 \cdots A_m)$ $(0 \leq A_i \leq 1)$ to a zone with k divisions (k is a multiple of m) is based on the following rule:

- For i $= 1 \rightarrow$ m, if $A_i < 0.5$, then i^{th} bit of the event $= 0$; otherwise, $=1$.
- For i $= m + 1 \rightarrow$ 2m, if $A_{i-m} < 0.25$ or $A_{i-m} =$[0.5, 0.75), then i^{th} bit of the event $= 0$, else 1.
- Repeat the same procedure until all k bits are assigned.

With the encoding rules for both zones and events, the next task is to route the event to the node that owns the zone (codeword) of the event. An example of inserting an event is illustrated in Fig. 2.5(a). The event with two attributes $\langle 0.8, 0.7 \rangle$ is routed to node 6 which owns the zone 1110. Similar encoding rules are applied to queries except that when the range of a query is larger than the range of a zone, it has to be divided into several sub-queries. An example of querying range event $\langle 0.5 - 1, 0.5 - 1 \rangle$ is illustrated in Fig. 2.5(b).

2.3.2 Classification of Data Gathering Mechanisms Based on the Direction of Diffusion

The data gathering process usually consists of two parts: query and reply. A sink (or user) sends a query to a sensor network and sensors that detect events matching the query send replies to the sink. Applications with different requirements opt for different communication paradigms. According to the direction of interest/data diffusion, there are three types of approaches [31]:

1. Two-phase pull diffusion: The most representative approach in this category is directed diffusion [35]. Both the queries for events of interest and the replies are initially disseminated via flooding and multiple routes may be established from a source to the sink. In the second pull phase, the sink reinforces the best route (usually with the lowest latency) by increasing its data rate (i.e., gradient). Data is then sent to the sink along this route. We will present the details of directed diffusion below. Two-phase pull diffusion is especially well-suited for scenarios with many sources and few sinks.

2. One-phase pull diffusion [31]: The overheads of flooding of both queries and replies are high in the case of a large number of sinks or sources. One-phase pull diffusion skips the flooding process of data diffusion. Instead, replies are sent back to neighbors that first send the matching queries. In other word, the reverse path is the route with the least latency. One-phase pull diffusion is also well- suited for applications with many sources and few sinks.

3. Push diffusion: In push diffusion, the roles of the source and the sink are reversed. A source actively floods the information collected when it detects an event and sinks subscribe to events of interest via positive enforcements. Push diffusion is well-suited for the two types of scenarios: (1) applications with many sinks and few sources and sources generate data only occasionally (2) mission-critical applications such as target tracking [94]. SPIN (Sensor Protocol for Information via Negotiation) [29, 44] can be classified as a protocol that realizes the notion of push diffusion. We will present the details of SPIN below.

Directed Diffusion: Directed diffusion [35] is a two-phase pull routing protocol in which data consumers (sinks) search for the data sources matching their interests and the sources find the best routes to route their data back to the subscribers. The procedures in directed diffusion can

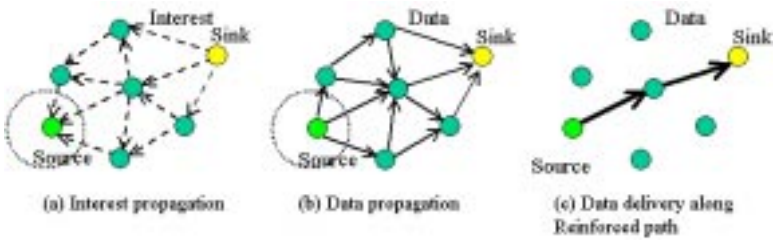


Figure 2.6: Three phases in directed diffusion [35].

be divided into three phases: interest propagation, data propagation, and reinforcement. Sinks first broadcast interest messages to their neighbors. When a node receives an interest packet, the interest packet is cached and rebroadcast to other neighbors if it is new to this node. The interest propagation also sets up the gradient in the network to facilitate data extraction to the sink. A gradient specifies both a data rate and a direction to relay data. The initial data rate of the gradient is set to be a small value and will be increased if the gradient along the path is enforced. When a node matches an interest (e.g., it is in the vicinity of the event in the target tracking application), it generates a data packet with the specified data rate. The data packet is unicast individually to the neighbors from whom the interest packet is received. When a node receives a data packet matching a query in its interest cache, the data packet is relayed to the next hop towards the sink. Both interest and data propagation are exploratory but the initial data rate is low. When a sink starts to receive data packets from some neighbors, it reinforces one of the neighbors by increasing the data rate in the interest packet. Usually such a neighbor is the one on the low-delay path. If a node receives an interest packet with a higher data rate then that in the interest cache, it also needs to reinforce the path. Since the entries in the interest cache are kept as soft state, eventually one of the path remains while other paths are torn down. The processes of three phases are depicted in Fig. 2.6.

SPIN [29, 44]: SPIN is a push diffusion protocol in which data sources initiate the data sending activities. SPIN consists of 3-stage handshaking operations (Fig. 2.7), including ADV (advertisement), REQ (request for data), and DATA (data message). Instead of directly flooding new data, the description of new data, i.e., meta-data, is exchanged in the first two advertisement-subscription phases to reduce exchange of redundant messages. If a node receives an advertisement with new information that is of interest to it, it replies with a request packet. The real data is then transmit-

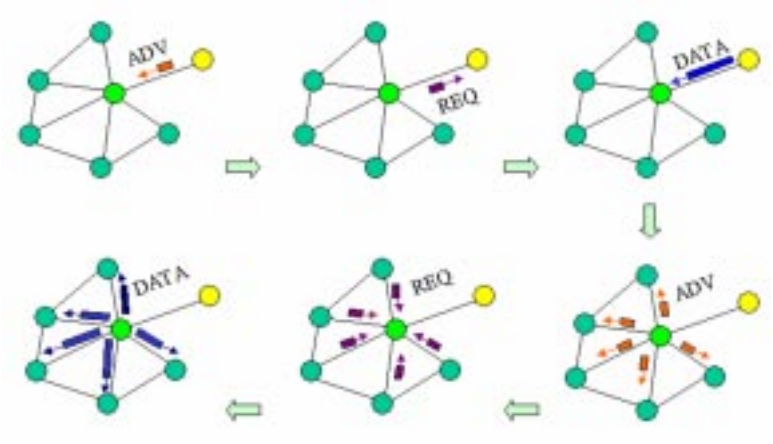


Figure 2.7: Three phases hand-shaking protocols in SPIN. [29]

ted in the third phase upon receipt of such a request. Propagation of new information is executed hop-by- hop throughout the entire network.

2.3.3 Classification of Data Gathering Mechanisms Based on the Structure of Dissemination

The number of sources and sinks in sensor network applications not only determines the direction of diffusion, but also plays a crucial role in laying the structure of dissemination in the system especially when it is considered in conjunction with data fusion. In what follows, we introduce four types of configurations, including tree, grid, cluster, and chain, and their representative approaches.

(1) ***Tree***: One of the most common dissemination structures used in sensor networks is one sink with multiple sources. A tree is usually rooted at the sink and spans the set of entire sources from which the sink will receive information. It is usually constructed in the reverse multicast fashion. TAG [54] and TinyDB [55] are two examples that use sink-tree based routing for data dissemination. On the other extreme of the scenario spectrum, there may be a single source and multiple sinks in certain applications. In this scenario, a tree is rooted at a source and constructed in the usual multicast fashion. The self-organizing multicast forwarding tree proposed by Mirkovic *et al.* [57] to disseminate reports from stimuli to multiple sinks falls in this category. The sinks broadcast their interest packets for certain events. Upon receipt of an interest packet, each sensor updates its distance to the sink and forwards the packet if it is new to the sensor. Each of the interest packets that record a minimum distance from some sink will be used by the source to construct the shortest path tree. The tree grows from the root and follows the reverse paths to

reach sinks. A new stimulus joins the tree via the closest sensor on the tree and creates a new branch of the tree. In Scalable Energy-efficient Asynchronous Dissemination protocol (SEAD) [41], a dissemination tree is built to deliver data from a source (root) to multiple mobile sinks (leaves). The tree is built upon an underlying geographical routing protocol. When a mobile sink would like to receive data from a source, it connects to the dissemination tree through one of its neighboring sensors, called as access node. Similar to the home agent in Mobile IP, the access node acts as an anchor node to relay data to the sink. When the sink moves out of the transmission range of its access node, it informs its access node of its new where-about by sending a *PathSetup* message. The latter will then forward all the information that the node is interested in receiving. When the distance to the original access node exceeds a pre-determined threshold, a sink joins a new access node. In order to reduce the number of messages transmitted over the tree, a source node duplicates its data at several replicas. The criterion for placing replica on the tree is to minimize the extra cost of constructing a branch for a new join request.

(2) ***Grid***: Similar to SEAD, Two-Tier Data Dissemination (TTDD) [89] is designed for scenarios with a single source and multiple mobile sinks. A grid structure rather than a tree structure is adopted as the dissemination structure in TTDD. An example grid structure originated from the source is shown in Fig. 2.8. In the higher tier, a source that detects an event proactively constructs a grid structure where sensors close to the grid points are elected as dissemination nodes. In the lower tier, a mobile sink sends a query to, and receives data from, its nearest grid point of the local grid. When a sink moves to another grid, it can quickly connect to the grid structure and the information access delay thus incurred is reduced. One of the applications for which TTDD is particularly well suited is target tracking in the battlefield.

(3) ***Cluster***: When data fusion is integrated with data dissemination, data generated by sensors are first processed locally to produce a concise digest which is then delivered to a sink. A hierarchical cluster structure [3, 30, 72] is better suited for this purpose. LEACH (Low-Energy Adaptation Clustering Hierarchy) [30] is a two-level clustering mechanism in which clusters are formed and sensors with sufficient energy may volunteer to become cluster heads (CHs) for carrying out in-network processing tasks. Once a sensor elects itself as the CH, it broadcasts a message to notify other sensor nodes of the fact that it is willing to be a CH. The remaining sensors then select a minimum transmission power to join their closest CHs. Within the cluster, a CH uses TDMA

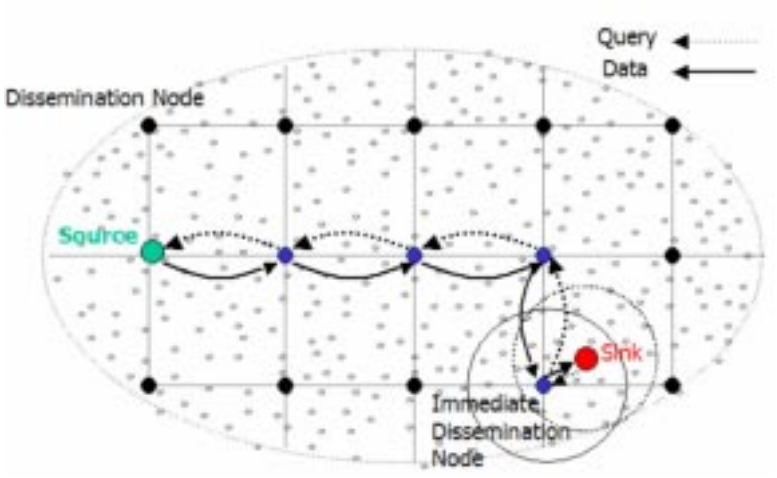


Figure 2.8: Two-Tier Data Dissemination (TTDD) Grid Structure. [89]

to allocate time slots to cluster members (so that the latter can relay their readings to the CH), compresses received data and transmits a digested report directly to the base station (sink). The role of a CH is rotated among sensors. Bandyopadhyay *et al.* [3] propose a multi-level hierarchical clustering algorithm. Similar to LEACH, this approach aims to realize the objective of balancing the load of sensors and achieving energy efficiency.

(4) ***Chain***: If energy efficiency and bandwidth usage is more important than the latency requirement, the chain structure that allows aggregation of data along a path ending at a sink is a competitive solution. **PEGASIS** (Power-Efficient GAthering in Sensor Information Systems) [50] is designed to aggregate data collected by all sensors in the entire network. Only one leader is elected each time and the leadership is rotated among all the sensors. Under the assumption that the network topology is a complete graph, the leader is able to connect all the sensors with the chain structure. Starting from the sensor at one end of the chain, data are propagated and aggregated along the chain towards the leader. Then the data dissemination and aggregation processes continue from the other end. The aggregations from both ends arrive at the leader, which directly transmit the aggregation result to the sink.

2.4 Data Fusion Mechanisms

As mentioned in Section 2.1 , in typical sensor network applications, sensors are deployed over a region to extract environmental data. Once data are gathered by multiple sources (sensors in the

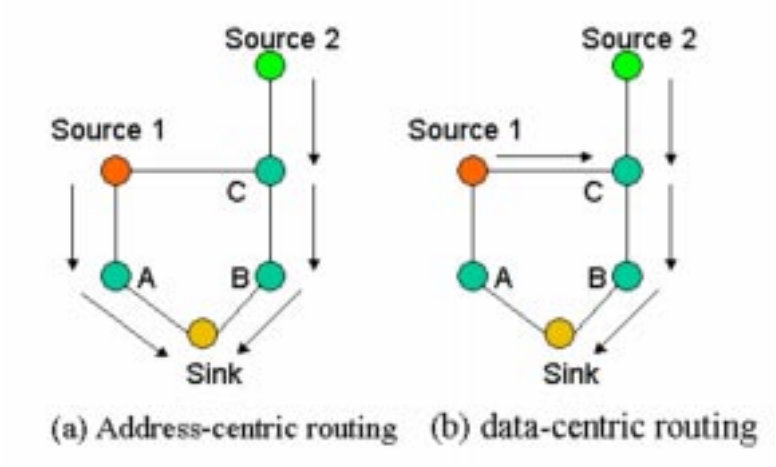


Figure 2.9: Address-centric routing vs. data-centric routing [42].

vicinity of the event of interest), they are forwarded perhaps through multiple hops to a single destination (sink). This, coupled with the facts that the information gathered by neighboring sensors is often redundant and highly correlated and that the energy is much more constrained (because once deployed, most sensor networks operate in the unattended mode), necessitates the need for data fusion. Instead of transmitting all the data to a centralized node for processing, data is processed locally and a concise digest is forwarded to other nodes or sinks. It reduces the number of packets to be transmitted among sensors and thus the usage in bandwidth and energy. The benefits of data fusion become obvious especially in a large-scale network. For a network with n sensors, the centralized approach takes $O(n^{3/2})$ bit-hops while data fusion takes only $O(n)$ bit-hops to transmit data [63]. When data fusion is considered in conjunction with data gathering and dissemination, the conventional address-centric routing, that finds the shortest routes from sources to the sink, is no longer optimal. Instead, data-centric routing, which considers in-network aggregation along the routes from multiple sources to a sink, achieves better energy and bandwidth efficiency, especially when the number of sources is large, and/or when the sources are located closely and far from the sink [42]. Fig. 2.9 gives a simple illustration of data-centric routing versus address-centric routing. Source 1 chooses node A as the relaying node in address-centric routing, but node C as the relaying and data aggregation node in data-centric routing. As a result, a smaller number of packets are transmitted in data centric routing.

Existing research activities of data fusion can be categorized into several groups:

1. Fusion function: Data fusion is generally applied for:

   a) Basic operations: the most basic operations for data fusion include: COUNT, MIN, MAX, SUM, and AVERAGE [54].

   b) Redundancy suppression: data fusion, in this case, is equivalent to data compression [16, 69].

   c) Estimation of a system parameter: Based on the observations from several pieces of sensor data, the data fusion function aims to solve an optimization problem to minimize the estimation error of a system parameter [63].

2. System architecture: Besides the sources and sinks, a sensor network that considers data fusion has one more component - the data aggregator. There exists a wide spectrum of ways to determine the location of the data aggregator.

3. Tradeoffs of resources: Depending on the resource constraints in the network, there exist the following tradeoffs: energy vs. estimation accuracy [8, 63], energy vs. aggregation latency [70, 92], and bandwidth vs. aggregation latency [69].

2.4.1 Classification of Data Fusion Mechanisms Based on Fusion Functions

The major purpose of incorporating data fusion into the data gathering and dissemination process is to reduce the number of packets to be transmitted and hence the energy incurred in transmission. There are two types of data aggregation: "Snapshot aggregation" is data fusion for a single event, such as tracking a target, while "periodic aggregation" periodically executes the data fusion function, such as monitoring an environment parameter periodically [8]. Depending on the application requirements, three types of data fusion functions can be used: basic aggregation functions, redundancy suppression, and estimation of a system parameter.

Basic Aggregation Function: The basic aggregation functions include five SQL-like operations: COUNT, MIN, MAX, SUM, and AVERAGE [54]. Here we use the structure of aggregates used in TAG and the AVERAGE function as an example to explain the procedures of aggregation. An aggregation component consists of three functions: a merging function f, an initializer i, and an evaluator e. The aggregation process starts with one sensor specifying the initial states for initializer

i, $\langle x, 1\rangle$, where the first entry in the 2-tuple is the sensor value of the starting node and the second entry represents the number of readings in the first entry. The aggregation packet including the initializer is propagated to the next hop and the merging function f is executed there for data aggregation. The merging function f is one of the five functions mentioned above and in the case of AVERAGE its function is expressed below:

$$f(\langle S_1, C_1\rangle, \langle S_2, C_2\rangle) = \langle S_1 + S_2, C_1 + C_2\rangle \tag{2.1}$$

which means that the first entry is the sum of sensor readings along the aggregation path and the second entry is the count of sensor readings. Finally, when the aggregation packet arrives at the sink (or the subscriber), the evaluator e calculates the final result $e(\langle S, C\rangle) = S/C$.

Although the basic functions share the same aggregation structure, the characteristics of different functions differ in three aspects (as summarized in Table 2.1 [54]):

1. *Duplicate sensitive*: Duplicate sensitivity indicates whether the result of the aggregation evaluator is affected by a duplicated reading from a single sensor. In the case of duplicate sensitive aggregates such as COUNT or AVERAGE, sending aggregation packets over multiple paths will lead to incorrect results.

2. *Exemplary or summary*: The result of *exemplary* aggregates might depend on any one value from the set of all sensor readings, while *summary* aggregates compute some property over all values. The results of exemplary aggregates are not predictable when one critical reading is lost.

3. *Monotonic*: The monotonic aggregates have a property that the aggregation over any combination of subsets of sensor readings will not affect the final result of aggregation. This property is an index of whether the evaluator function can be applied multiple times in networks before the final evaluation.

Redundancy Suppression: Due to the fact that correlation exists in sensor data both in the spatial and temporal domains, one of the most important data fusion functions to reduce the message overhead is to eliminate data redundancy (or equivalently exploit the correlation structure that exists in sensor data) via distributed source coding. Chou *et al.* [16] propose to incorporate both

	MAX, MIN	COUNT, SUM	AVERAGE	MEDIUM
Duplicate Sensitive	No	Yes	Yes	Yes
Exemplary(E)/Summary(S)	E	S	S	E
Monotonic	Yes	Yes	No	No
Partial State	Distributive	Distributive	Algebraic	Holistic

Table 2.1: Classes of aggregation functions [54]

distributed source coding and adaptive signal processing in data fusion and exploit the correlation structure in sensor data to save energy incurred in transmission. The system architecture consists of a data gathering node (sink) and sensors. The data gathering node sends queries to sensors sequentially to obtain certain information that pertains to the entire field. The design objective here is to devise a computational inexpensive encoding operation supporting multiple compression rates in the sensors while allowing a more complex decoding procedure in the data gathering node. The authors leverage the Slepian-Wolf theorem [75] as the theoretical base: if two discrete random variable $\mathbf{X}$ and $\mathbf{Y}$ are correlated, then $\mathbf{X}$ can be losslessly compressed using $\mathbf{H(X|Y)}$ bits without access to $\mathbf{Y}$, where $\mathbf{H(X|Y)}$ is the conditional entropy of $\mathbf{X}$ given the information of $\mathbf{Y}$. Then they propose a blind compression method to achieve the theoretical bound given in the Slepian-Wolf theorem. Suppose that the data gathering node has full information of $\mathbf{Y}$ (side information) and the difference between $\mathbf{X}$ and $\mathbf{Y}$ is less than $2^{i-1}\Delta$. With their proposed tree-based codebook, one can encode $\mathbf{X}$ with only i bits without any knowledge of $\mathbf{Y}$ and the decoder can fully recover the information of $\mathbf{X}$. An example that shows the encoding and decoding operations with the tree-based codebook is depicted in Fig. 2.10. If the sink has collected the full information of $\mathbf{Y}$ (e.g., $\mathbf{Y}$=0.4) and that the difference between the value of $\mathbf{X}$ and $\mathbf{Y}$ is less than 0.2, then under the case that the sampling resolution, Δ, is equal to 0.1, only 2 bits are required to encode the value of $\mathbf{X}$. The deterministic encoding function is: *f(X) = index(X)* mod 2^i, where the index function converts the value to the index in the tree-based codebook, e.g., f(0.5) = 5 and $\mathbf{X}$ is encoded in 2 bits as 01 because 5 mod 4 equal to 1. After the sink receives the i-bit encoding information of X, it traverses the tree and finds the corresponding subcodebook S. In the given example, $S = \{r_1, r_5\}$. The decoding rule is: $\hat{X} = \arg\min_{r_i \in S} \|Y - r_i\|$. Therefore, the sink can decode the value of $\mathbf{X}$ to be 0.5.

The procedures to obtain the correlation structure of sensor data can be divided into two steps. First the data gathering node collects the uncompressed data from all sensors in the first

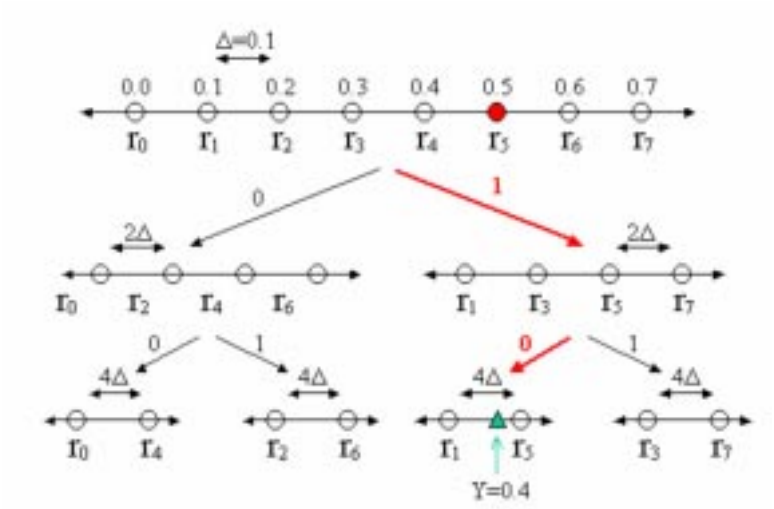


Figure 2.10: An example of the tree-based codebook [16].

K rounds to obtain the temporal and spatial redundancy in data. Then in the following round of data collection, an adaptive filtering framework is used at the data gathering node to learn the correlation structures in the data.

Duarte-Melo *et al.* [20] address the issue of joint design of data compression and data dissemination. They consider the same system architecture as [16] (i.e. the system consists of a sink and multiple sensors), and formulate the problem as a non-linear programming problem that maximizes the system lifetime, subject to the constraints on flow conservation, energy, and sampling rates. The last constraint specifies the least sampling rate for Slepian-Wolf type of encoding. Scaglione and Servetto [69] propose to integrate routing with source coding under a system in which each sensor reports its data to the entire network and all sensors intend to capture certain information that pertains to the entire field within a prescribed distortion value, i.e., the joint entropy of all readings. As the data is propagated to a node, it is encoded with the local data, and the compressed data is relayed to the next hop node. It is shown that the aggregation method consumes bandwidth in a scalable way (i.e., below the transport capacity) and thus the problem of vanishing per-node throughput [27] is avoided.

Petrovic *et al.* [61] propose the Data Funneling mechanism. In Data Funneling, multiple reports from different sensors are sent to the sink at approximately the same time. Since these packets have similar headers, they can be merged to a single packet by removing their redundant headers. Considerable saving can be made by a simple concatenation of readings in the packet body. A source

coding based on the ordering is further applied to compress the data of concatenated readings. For instance, suppose all the readings are integers and their range is in $[0, 1, \cdots, 5]$, i.e., there are six possible values for a reading. Then we can compress the data set with four readings by simply sending three readings. The fourth reading is implicitly encoded by the order of the three readings because there are six (3!) combinations of the ordering relationship.

Estimation of a system parameter: One of the main objectives of gathering an enormous amount of data in sensor networks is to estimate certain parameter in the environment based on the collected data. In this category of applications, sensors cooperate to disseminate necessary information to certain nodes which then proceed to estimate the parameter of interest. The estimation problem can be formulated as an optimization problem, with the objective of minimizing the estimation error. An example of such an optimization problem is to average all the temperature readings of sensors within a room to estimate the temperature of a room. The estimation is optimal with respect to the minimum square error (MSE) criterion. Another example is to track targets to minimize the estimation error of the target's location. Rabbat and Nowak [63] formulate an optimization problem to estimate a system parameter, θ, given a set of data from n sensors, $\mathbf{x}$. The objective is to minimize the cost function f :

$$\min_{\theta} f(\theta, \mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} f_i(\theta, x_i) \tag{2.2}$$

The gradient decent method is one of the most popular techniques for iteratively solving the optimization problem. It can be applied at a central server with the entire set of data:

$$\hat{\theta}^{(k+1)} = \hat{\theta}^{(k)} - \alpha \sum_{i=1}^{n} g_{i,k} \tag{2.3}$$

where $g_{i,k} \in \partial f_i(\hat{\theta}^{(k)})$, α is a small positive step size, and k is the iteration number. In [63], Eq. 2.2 is solved with a decentralized incremental approach by dividing Eq. 2.3 into a cycle of n subiterations. Each subiteration focuses on optimizing a single component, $f_i(\theta)$, at a node based on its local data. Same as **PEGASIS**, the task of subiterations is rotated hop by hop in a chain. The subiteration starts from a node which inherits the estimation result from the previous iteration, $\varphi_0^{(k)} = \hat{\theta}^{(k-1)}$, where $\varphi_i^{(k)}$ is the result of the i^{th} subiteration within the k^{th} iteration. The task of

each subiteration executed at each node is:

$$\varphi_i^{(k)} = \varphi_i^{(k)} - \alpha g_{i,k} \quad i = 1, \cdots, n \tag{2.4}$$

Finally the estimation in the k^{th} iteration ends at the n^{th} subiteration: $\hat{\theta}^{(k)} = \varphi_i^{(k)}$. No more than $O(\varepsilon^{-2})$ iterations are required for the desired accuracy ε. Three applications, robust estimation, localization, and clustering and density estimation, are illustrated that leverage the distributed optimization in [63].

Zhao *et al.* [94] propose a leader-based tracking scheme in which samples are collected successively at different time instants and locations to localize the target. A sensor that contains the most information is elected by the previous leader to estimate the current location of the target based on the past belief and the current measurement. Similar to the directed diffusion approach [35], a routing protocol called constrained anisotropic diffusion routing (**CADR**) [17] is used to redirect queries from users to the most qualified leader. The current leader adopts the sequential Bayesian filtering technique to estimate the current location of the target. A sensor, which is estimated to hold the maximum information, is then chosen to be the next leader. This process is termed as information-driven sensor query (**IDSQ**).

2.4.2 Classification of Data Fusion Mechanisms Based on System Architecture

Data Funneling [61] is intrinsically an energy efficient routing protocol [71] integrated with data aggregation and compression techniques. The basic idea of data funneling is to build a cost field with the funnel shape to pull the data from sources to the sink. The sink initials directional flooding to send an interest packet towards a target region as shown in Fig. 2.11(a). During the forwarding process of interest packets, a forwarder computes its (energy) cost for communicating back to the sink and updates the cost field in the interest packet. When a node within the target region receives an interest packet from the nodes outside the region, it designates itself to be a border node. The cost required to reach the sink (i.e., the cost in the interest packet) is recorded, and moreover a field that is used to keep track of the cost to reach the border node is included. Since there could be multiple "entries" (border nodes) to the target region, a node within the target region might receive multiple interest packets from border nodes. Instead of requesting all the nodes to send individual reports back to the sink, one of the border nodes is responsible for the task of collecting

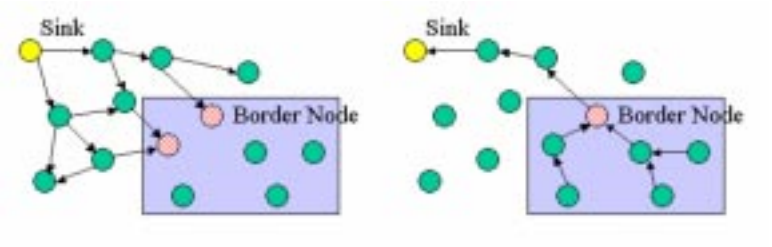


Figure 2.11: Data Funneling (a) Direction flooding phase (b) Data communication phase [61].

and aggregating all reports in the region and sending a single packet to the sink. All the sensors within the region share a common schedule of which border node to be the data aggregator during each round of reporting. The schedule is determined by a deterministic function of the costs to reach the sink from all border nodes. Sensors with a longer distance to the designated aggregator send their reports earlier and the readings are concatenated in a single packet to eliminate redundant headers. The data communication process is shown in Fig. 2.11(b). After receiving reports from all the sensors within the region, the designated border node further compresses the data by applying a coding technique based on ordering.

DFuse [45] is a distributed data-fusion framework especially designed for video streaming applications. The framework provides the flexibility of data fusion in two aspects. First, a layer of fusion modules provides a set of data fusion functions for an application to manage video streams. Second, the role of a node (a sink, a relay or a fusion point) is determined in a distributed way based on the given cost function.

Baek *et al.* [2] study optimal data fusion strategies regarding the order of fusion and the organization of fusion devices under two scenarios: networks with a single sink and those with multiple sinks. In the case of a single sink, all sensors send their non-redundant data to the sink. The optimal fusion strategy is to determine the order of compression at each node, so as to minimize the overall energy consumption while faithfully disseminating the data from all sensors to the sink. That is, the sum of rates for any subset of sensors is lower bounded by the conditional entropy, given that the sink has known the data of the rest set of sensors (based on Slepian-Wolf theorem [75]). The optimal solution can be found using a greedy algorithm. The sensor with the least communication cost to reach the sink first transmits its data without compression to the sink. Then in the increasing order of communication costs to reach the sink, sensors sequentially

disseminate their compressed data given the known side information at the sink. Surprisingly, the optimal solution is independent of the correlation structure of the data, and simply relies on the topology of the network. In the case of multiple sinks, a three-level hierarchical architecture that includes sensors, compressors, and sinks is considered. Both compressors and sinks can aggregate data from sensors with a compression ratio α ($0 < \alpha < 1$). All the compressed data is destined for any one of sinks. Therefore, a sensor transmits its raw data either directly to its closest sink or a nearby compressor, which compresses the data and forwards the compressed data to its closest sink. Baek *et al.* show that under a given α, the optimal organization that minimizes energy consumption is a Johnson-Mehl tessellation in which the entire sensing field is divided into regions either belonging to a sink or a compressor. In the two extreme cases that α is equal to 0 and 1, representing full and none compression, the optional organization degenerates from Johnson-Mehl tessellation into Voronoi tessellations that constitute the set of sinks and compressors and the set of sinks, respectively.

2.4.3 Classification of Data Fusion Mechanisms Based on Tradeoffs in System Resource

Depending on the resource constraints, there exist various tradeoffs in different data fusion schemes: energy versus estimation accuracy [8, 63], energy versus aggregation latency [70, 92], and bandwidth versus aggregation latency [69].

Tradeoff between energy and accuracy: With respect to the tradeoff between energy consumption and accuracy of aggregation results [8, 63], the requirement of higher accuracy demands more message exchanges and leads to higher energy consumption. In [8], a distributed periodic aggregation approach is proposed to estimate the maximum of sensor data in a field, where the maximum of sensor data is modeled to be Gaussian distributed. Compared with multiple "snapshot aggregations", the proposed approach exploits the energy-accuracy trade-off, and provides users with a system-level knob to control the desired accuracy and energy consumption. The distributed optimization approach proposed in [63] shows that $O(\varepsilon^{-2})$ iterations of aggregation are required to achieve the desired accuracy ε. Similar to [8], model based query is supported in [19], i.e., a declarative query processing engine uses a probabilistic model to answer questions about the current state of the sensor network. The model is based on time-varying multivariate Gaussians.

The initial model is constructed based on the historical data and updated when new data are available. Given a query specifying the confidence interval, the problem is to choose the best set of new observations such that the cost of collecting new data is minimized.

Ye *et al.* [91] figure robustness of delivery in the design. Multiple, interleaved paths between the source and the sink enable the network to be more resilient to node or transmission failure. In their proposed GRAdient Broadcast (GRAB) protocol, the cost field is constructed first. The cost of a node represents the minimum energy required to forward a packet along a path to the sink. To exploit the redundancy of delivery, a sender (forwarder) broadcasts a packet, and a relay node forwards the packet only if its cost is smaller than the sender's cost. Moreover, the degree of delivery redundancy is controlled, as paths are expanded quickly from the source, maintained within a reasonable width next, and finally shrunk near the sink.

Tilak *et al.* [77] trades accuracy of information for energy saving in data dissemination. They advocate non-uniform information granularity, i.e., the required accuracy of information decreases as the distance from the source becomes longer. Applications in battlefield or disaster rescue scenarios usually possess such characteristics. Two deterministic and two non-deterministic protocols are designed for non-uniform information dissemination. The proposed protocols trade accuracy of information for energy expenditure by selectively discarding packets from a sensor.

Tradeoff between energy and latency Both Schurgers *et al.* [70] and Yu *et al.* [92] explore the tradeoff between energy consumption and propagation latency from data sources to the sink, but from different perspectives. In [70], energy is saved via directly turning off the radio circuitry when a sensor is not transmitting or receiving data. While the low duty cycle operation reduces power consumption in idle state, it increases the propagation latency from the data source to the sink. The protocol, Sparse Topology and Energy Management (STEM), is proposed to deal with the problem. STEM utilizes dual bands for data transmission and wakeup signaling. The channel for wakeup signaling is operated in a low duty cycle. Each node periodically turns on the radio circuitry for the wakeup channel to listen whether any other node has attempted to communicate with it. Once a node detects such an activity in the wakeup signaling channel, it turns on its radio circuitry for the data channel. The increased latency due to the sleep state is thus bounded by the sleep-listen period in the wakeup channel. STEM is especially well-suited for applications with most operations in the

monitoring state. For instance, in a fire alarm system, the network only senses the environment in an energy- efficient way and the system stays in the monitoring state most of the time. Once an event takes place, the system changes to the transfer state quickly and reports the event to the data sink in a timely manner. Yu *et al.* [92] achieves energy saving via adjusting the modulation scaling factor, i.e., the number of bits in a modulated symbol. In general, a smaller modulation scaling factor reduces the power required in the transmission, but increases the transmission time over a link although the relationship between the power required and the link delay thus increased is not necessarily monotonic. The authors consider a multiple-source single sink data aggregation tree, and formulate the problem of finding an optimal schedule of packet transmission to minimize the total transmission energy incurred at all nodes in the aggregation tree, subject to the given propagation latency constraints. A numerical optimal algorithm, a pseudo-polynomial, dynamic programming based approximation algorithm and a distributed on-line protocol are developed to solve the problem.

Tradeoff between bandwidth and latency Scaglione and Servetto [69] discuss the tradeoff between the bandwidth usage and the decoding delay. They argue that data aggregation along a path leads to better bandwidth usage, but if the aggregation is conducted along multiple parallel paths, the delay incurred in aggregating and sending data to destinations is reduced but the bandwidth usage (or the corresponding energy consumption) is increased. They then suggestion that these two quantities are linked together by the routing strategy applied.

2.5 Target Tracking System and Clustering

The requirements for target tracking systems are usually different from those of general monitoring systems, especially in latency and bandwidth usage. In Chapter 3, we propose a dynamic clustering algorithm for acoustic target tracking system, a special case of data fusion mechanisms designed for efficient coordinations of message exchanges. In this section we introduce the related work of both acoustic tracking approaches and clustering mechanisms proposed by other researchers.

Work on the use of wireless sensor networks for acoustic tracking Use of wireless sensor networks for acoustic tracking has been investigated by quite a number of researchers and existing work can be roughly categorized into two categories. Research in the first category aims to achieve

collaborative processing. A sufficient (and sometimes significant) amount of data are collected by sensors and processed at a coordinator (or a CH in our context). Maximum likelihood testing (ML) [73] and minimum square estimation [46] are two representative techniques to localizing the position of the target. Brook *et al.* [9] present a self-organized distributed tracking framework in which three methods, namely pheromone routing, extended Kalman filter, and Bayesian entity tracking, are applied. Approaches in this category usually give accurate results but at the expense of heavy message exchanges.

Instead of collecting samples at the same time from different sensors, approaches in the second category use samples collected successively at different time and locations for acoustic tracking. In the leader-based tracking scheme proposed by Zhao *et al.* [94, 17, 51], a sensor that contains the most information is elected by the previous leader to estimate the current location of the target based on the past belief and current measurement. Similar to the directed diffusion approach [35], a routing protocol called *constrained anisotropic diffusion routing (CADR)* [17] is used to redirect queries from users to the most qualified leader. The current leader adopts the sequential Bayesian filtering technique to estimate the current location of the target. A sensor, which is estimated to hold the maximum information, is then chosen as the next leader. This process is termed as *information-driven sensor query (IDSQ)* [94]. Because IDSQ is effective only for single target tracking, a group formation algorithm is proposed in [51] to handle multiple targets. A *suppression* message sent from the current leader is used to force neighboring sensors to abandon detection and join the group. Note that the main purpose of group formation here is to suppress contention, but not for collaborative information processing (as in the proposed dynamic clustering approach). In spite of the little message overhead incurred, these approaches may not be able to achieve high accuracy, as they do not exercise collaborative information processing.

The proposed dynamic clustering approach lies somewhat between the first and second categories. An adequate amount of data is collected and processed at CHs. By "adequate," we mean that the location information will be gathered from a sensor node S_j with the largest detected signal strength, and all of S_j's Voronoi neighbors. With the use of Voronoi diagram, we can achieve high accuracy without incurring excessive message overhead. Moreover, unlike the passive, query-based scheme such as IDSQ/CADR, CHs in our proposed scheme report their tracking results to users in a proactive way so as to reduce the latency. We believe this proactive structure is better suited for

a system that has only a few query users and demands fast responses. Finally, the proposed scheme can support and handle the multiple target tracking case in a straightforward manner — as each information solicitation packet contains the signal signature, a sensor that detects multiple targets can report appropriately to different CHs (that become active in response to different acoustic targets).

Work on clustering techniques The clustering technique has been extensively exploited to resolve the scalability and energy conservation issues in sensor networks. Heinzelman *et al.* [30] propose the *LEACH* mechanism in which one-level clusters are formed by sensors which volunteer to become the CHs. The task of being a CH is rotated between sensors. Once a sensor elects itself as the CH, it broadcasts a message. The remaining sensors join their closest CH by selecting the minimum transmission power. Bandyopadhyay *et al.* [3] propose a multi-level hierarchical clustering algorithm. Both approaches aim to realize the objective of balancing the load of sensors and achieving energy efficiency. They may not be directly applicable to target tracking because (i) all the elected CHs may not be close enough to the target and (ii) clusters are not formed uniformly such that a CH may not recruit a sufficient number of sensors. Moreover, the issue of mitigating contention between clusters is not addressed in both approaches.

Ye *et al.* [89] has also proposed a hierarchical model, called the *two-tier data dissemination (TTDD)* model. In the higher tier, a source sensor which detects the target proactively constructs, for the purpose of routing, a grid structure so that the tracking information can be effectively disseminated throughout the entire system. In the lower tier, a mobile sink retrieves the tracking information from the nearest grid point of the local grid. The TTDD model is especially well-suited for the case of multiple mobile sinks. TTDD and our proposed work share the similarity of dynamically constructing grids/clusters in response to tracking targets, but differs in that the grid structure laid in TTDD is not for the purpose of collecting sensor information and thus does not consider issues (**I2**) and (**I3**) as outlined in Chapter 1.

It has also come to our attention that two clustering approaches for target tracking have been proposed recently [87, 93]. Zhang *et al.* proposed the *DCTC* algorithm [93] in which a tree-structured cluster is formed when a target is detected in the sensing field. The sensor closest to the target is selected to be the root (or the CH in our context) of the tree through explicit leader

election. Then, a minimum cost tree that spans all the sensors within a certain range from the target is constructed. Reconfiguration of the tree is triggered when the distance from the target to the root exceeds a pre-determined threshold. There are two potential problems in DCTC. First, the cost of reconfiguring a tree, including election of a new root and expanding and pruning of the tree, may be very high. In contrast, the cost of reconfiguring a cluster in the dynamic clustering approach is much lower. Second, the formation of a tree relies heavily on the knowledge of the target position which cannot be obtained *after* the root collects data from sensors. Yang *et al.* [87] propose to construct static clusters. A cluster is activated upon detection of a target. Using a similar idea in [94], the CH of the currently activated cluster uses linear prediction result for the target position to determines whether the tracking task needs to be switched to another CH. Unlike [87], the proposed dynamic clustering approach is stateless and does not require information of the past history to determine the target position. As a result, it is more robust when the paths along which targets travel are unpredictable. On the other hand, the prediction techniques can be incorporated to the proposed approach to further reduce the communication overhead.

2.6 Optimization in Data Gathering

One special category of data gathering approaches is to treat the problem of data transport in sensor networks as an optimization problem whose objective function is to maximize the amount of information (utility) collected at sinks, subject to various kinds of constraints, including flow, energy, latency, channel bandwidth constraints, and etc. Such optimization problems encompass both flow control problem and routing problem. They intend to solve simultaneously the problems of maximizing information throughput and mitigating congestion. Furthermore, the data gathering and fusion approaches presented previously can be incorporated with the optimization approaches in a complementary manner. The optimization of data gathering can be formulated as a non- convex programming problem solved in a centralized manner for the evaluation of the performance bound or a convex programming problem solved in a distributed way for practical usage. A non-convex programming problem formulation general enough to encompass a wide variety of applications in sensor networks, each with a different objective function and subject to different constraints, will be introduced in Chapter 4. In this section various simplified convex or linear programming problems are discussed.

The optimization problem formulations in sensor networks are application-oriented as the characteristics and requirements of applications are different. Specifically, there are three types of the objective functions in the optimization problem: (1) maximizing the overall utility of sensor data collected at sinks [10], (2) maximizing data extraction from sensors [65, 58], and (3) maximizing the system lifetime (or equivalently minimizing the energy expenditure) [5, 58]. Byers *et al.* [10] consider the optimization problem of maximizing the overall utility of sensor networks during the system lifetime, subject to an energy constraint. The energy constraint is expressed as a high level cost on sensing, transmission, reception, and aggregation. Chang *et al.* [11] devise a routing solution to maximize the system lifetime of sensor networks under the given source rate of nodes and subjected to flow conservation and energy constraints (only considering the transmission power consumption). Without considering the node capacity constraint, the problem was reduced to a linear programming problem. Sadagopan *et al.* [65] use an iterative approximation algorithm to solve a similar linear programming problem except by changing the objective function from maximizing system lifetime to maximizing data extraction. Duarte-Melo *et al.* [20] propose a nonlinear programming jointly considering data compression and data dissemination problem to maximize the number of snapshots generated from the networks, or equivalently maximize the system lifetime. The goals of maximizing information extraction and minimizing energy consumption always conflict and could not be satisfied at the same time. Ordonez and Krishnamachari [58] consider the scenario of single sink and multiple sensors and two kinds of nonlinear optimizing problems are discussed: one is for maximizing the total information gathered subject to an overall energy constraint and the other complementary model is for minimizing the overall energy consumption subject to minimal information rate requirements. Other constraints considered include fairness constraints, in which the flow rate from one sensor is restricted to be less a portion of the overall rate arrived at the sink, and channel capacity constraints based on Shannon's theoretical capacity bound by assuming an interference- free communication model. These two complementary models are shown to be equivalent in terms of a correspondence between optimal solutions and constraints.

2.6.1 Utility-Based Approach

Utility based approaches have been exploited in conventional wired networks (e.g., [40, 52]), cellular wireless networks (e.g., [67]), ad hoc networks (e.g., [62, 86]), and most recently sensor networks

[10]. Kelly *et al.* [40] propose a pricing scheme to achieve weight proportional fair rate allocation for users in the wireline environment. The same problem considered in [40] is solved by Low *et al.* [52] differently such that the dual problem can be optimized in a distributed manner. Both Xue *et al.* [86] and Qiu *et al.* [62] extend Kelly's work [40] and consider the rate allocation problem in ad hoc networks. The major differences between Xue *et al.* [86] and Qiu *et al.* [62] lie in that (i) the former [86] uses the link capacity as the constraint of the channel capacity, while the latter [62] uses the node capacity as the constraint; and (ii) while the formulations in both work [86] and [40] divide the system problem into the user and network problems, the work reported in [62] incorporates the forwarding cost in the user optimization problem. None of the work in [40, 52, 86, 62] consider the energy constraints which we believe is one of the most important criteria in sensor networks.

Saraydar *et al.* [67] take a utility based approach to control transmission power in a decentralized manner in a multi-cell wireless data system. Recently Byers *et al.* [10] consider the optimization problem of maximizing the overall utility of sensor networks during the system lifetime, subject to an energy constraint. The energy constraint is, however, expressed as a high level cost, and does not differentiate power consumed in transmission, reception, and idle states. Chang *et al.* [11] devise a routing solution to maximize the system lifetime of sensor networks. As neither the link capacity nor the node capacity is considered in their work, the solution thus derived may not be feasible. Sadagopan *et al.* [65] solve a similar linear programming problem with an iterative approximation algorithm. Also, none of the existing work differentiates the treatment of packets with respect to their quality or information. In contrast, our proposed approach not only considers the energy constraint but also differentiates treatment of packets with respect to their quality of information.

In the above existing work, optimal flow control problem and routing problem are considered separately. Routes of flows are determined first by a routing algorithm and then considered as invariant settings in the flow optimization problem. Wang *et al.* [81] show that the optimization problem considering both routing and flow control decisions together is a NP-hard problem. They also show there exists a tradeoff between utility maximization and route instability. The routing metric based on pure dynamic pricing information achieves high utility but results in instable routing. By adding static component such as hop count to the routing metric stabilizes routing decisions. Kar *at. al* consider the flow optimization problem along with multi-path routing. In

general, the minimum price among all the paths from a sender determines the rate of the sender.

As compared with the aforementioned utility-based approaches, our proposed approach fixes problems of applying utility based approaches to wireless sensor networks in three directions: on-line estimation of node capacity (to be used in the capacity constraint of the optimization problem) in the multi-hop wireless environment, inclusion (and linearization) of energy constraints that relate the system lifetime to the data rate, and incorporation of optimization results in selecting routes that maximize the utility.

2.6.2 Transport Control in Sensor Networks

Besides the utility-based approach, there exist many other transport control protocols designed for wireless sensor networks to provide two main functions in the transport control layer, i.e. congestion control and end-to-end reliability, including:

1. *PSFQ*[79]: *PSFQ* focuses on the reliable transport operations, which are necessary under certain scenarios, including network reprogramming, debugging, and critical commands. *PSFQ* consists two operations: (i)*pump slowly*: periodically a packet is generated from the source in a low rate so that the network is not congested and (ii) *fetch quickly*: once a node detect a packet loss from the discontinuity of sequence number, it requests a retransmission from its previous hop immediately.

2. *CODA*[80]: Different from *PSFQ*, the goal of *CODA* is simply congestion detection and avoidance. It consists of three mechanisms: (i) congestion detection via buffer status and observation of channel loading, (ii) open-loop hop-by-hop backpressure, and (iii) closed-loop multi-source regulation.

3. *ESRT*[66]: ESRT intends to provide both reliability and congestion control although the reliability defined in *ESRT* is the ratio of the number of received packet to the number of the packets required for reliable event detection. The reliability increases when the source increases the data rate under the critical threshold of congesting the network. On the other hand, once a forwarder detects the network congestion, it sets the congestion notification bit in the packet. When the sink receives a packet with the congestion bit being set, it broadcast a congestion notification (which is assumed to be able reach the entire network within one

hop transmission) to decrease the data rate of sources.

4. *Speed*[28]: The goal of *Speed* protocol is to disseminate the data from the source to the destination at a steady *speed*. That is, the end-to-end propagation delay is proportional to the distance from the source to the destination. Similar to *CODA*, *Speed* has both open-loop and closed-loop control mechanisms, including back-pressure rerouting and neighborhood feedback loop, respectively.

5. [83]: In [83] Woo *et al.* propose a transmission control mechanism at both MAC and transport layers. Instead of providing reliability or congestion control services, their proposed approach focuses on the fairness issue for the networks with a sink tree.

There exists both similarities and differences in between the proposed utility-based approach and the work above. On-line measurements based on local information are used to evaluate the channel condition. Nevertheless, the main difference is that the above approaches do not emphasize on the data-centric operations.

Chapter 3

Dynamic Clustering for Acoustic Target Tracking System

In the applications of tracking systems, the localization of targets needs the data from sensors at different locations. Because of the limited available bandwidth and the requirement of reporting the locations of targets to the subscribers (usually remote controllers) in a timely manner, the method of collecting and processing data at the central server is infeasible for tracking systems. Instead, it is desirable to have sensors collaborate on processing the data and transporting a concise digest to subscribers. This reduces not only the number of packets to be transported, but also the probability of collision and interference in the shared media. *Localized and collaborative* data processing also aids in reducing the power consumed in communication activities and hence prolonging the lifetime of sensor networks.

To facilitate collaborative data processing in target tracking-centric sensor networks, the cluster architecture is usually used in which sensors are organized into clusters, with each cluster consisting of a cluster head (CH) and several neighboring sensors (members). In the conventional cluster architecture, clusters are formed statically at the time of network deployment. The attributes of each cluster, such as the size of a cluster, the area it covers, and the members it possesses, are static. In spite of its simplicity, the static cluster architecture suffers from several drawbacks. First, fixed membership is not robust from the perspective of fault tolerance. If a CH dies of power depletion, all the sensors in the cluster render useless. In the case that one or more sensors die, a cluster may not have sufficient sensors to carry out its tracking tasks. Second, fixed membership prevents sensor nodes in different clusters from sharing information and collaborating on data processing. Lastly, fixed membership cannot adapt to highly dynamic scenarios in which sensors in the region

of high (low) event concentration may be instrumented to stay awake (go to sleep).

Dynamic cluster architectures, on the other hand, offer several desirable features. Formation of a cluster is triggered by certain events of interest (e.g., detection of an approaching target with acoustic sounds). When a sensor with sufficient battery and computational power detects (with a high signal-to-noise ratio, SNR) signals of interest, it volunteers to act as a CH. No explicit leader (CH) election is required, and hence no excessive message exchanges are incurred. As more than one "powerful" sensors may detect the signal, multiple volunteers may exist. A judicious, decentralized approach has to be applied to ensure that only one CH is active in the vicinity of a target to be tracked with high probability. Sensors in the vicinity of the active CH are "invited" to become members of the cluster and report their sensor data to the CH. In this manner, a cluster is only formed in the area of high event concentration. Sensors do not statically belong to a cluster, and may support different clusters at different times. Moreover, as only one cluster is active in the vicinity of a target with high probability, redundant data is suppressed and potential interference and contention at the MAC level is mitigated.

In this chapter, we devise and evaluate a fully decentralized, light-weight, dynamic clustering algorithm for single target tracking. We focus on acoustic target tracking, although the proposed approaches can be readily applied to other types of tracking applications. Sensors in the acoustic tracking systems perform two types of computation: (1) sensing the energy level of signals; and (2) analyzing and classifying the sound, and performing the data fusion. The former is not computational intensive and can be handled by sensors with minimal computation power. The later, however, requires much higher computation power. To this end, we envision a hierarchical sensor network that is composed of (a) a static backbone of sparsely placed high-capability sensors which will assume the role of a CH upon triggered by certain events of interest; and (b) moderately to densely populated low-end sensors whose function is to provide sensor information to CHs upon request. A cluster is formed and a CH becomes active in an on-demand fashion, when the acoustic signal strength detected by the CH exceeds a pre-determined threshold. The active CH then broadcasts an information solicitation packet, asking sensors in its vicinity to join the cluster and provide their sensing information. After receiving a sufficient number of replies from sensors, the CH applies a localization method to estimate the location of the target and send a report to the subscribers. In acoustic tracking, there are two common methods for target localization: delay-based [82] and

energy-based [46, 73]. We adopt in this chapter the energy-based localization method and will show that it is robust in the presence of moderate noise and movement of the targets.

There are several issues which we have to address and devise solution approaches for, in order to realize the notion of dynamic clustering: (**I1**) how CHs cooperate with one another to ensure that only one CH (preferably the CH that is closest to the target[1]) is active with high probability; (**I2**) when the active CH solicits for sensor information, instead of having all the sensors in its vicinity reply, only a sufficient number of sensors respond with non-redundant information; and (**I3**) both the packets that sensors send to their CHs and packets that CHs report to subscribers do not incur significant collision.

To deal with these issues, we propose, with the use of Voronoi diagram, a probabilistic leader volunteering procedure and a sensor replying method. Initially, we enable all the sensors to calibrate their relative positions to their neighbors (at the CH $\leftrightarrow$ CH level and the sensor $\leftrightarrow$ sensor level) at the time of network deployment. Then, with the use of Voronoi diagram, each CH (or sensor) can calculate and tabulate the probability that given an distance estimate between a target and itself, the CH (sensor) is closest to the target. This information is used to set up the back-off timer used by a CH to announce its willingness to be active in the leader volunteering process. If no other CHs volunteer before the timer expires, the CH becomes active; otherwise, it suppresses its timer. Similarly, this probabilistic information is also used by a sensor to determine whether or not it should respond to a CH's request and the corresponding back-off timer value. Note that due to the limited mobility nature of sensors the calibration and tabulation process needs only to be carried out once at network deployment and rather infrequently during system operation.

The rest of the chapter is organized as follows. We introduce the necessary technical background and give an overview of the proposed dynamic clustering algorithm in Section 3.1. We delve into the algorithm details in Section 3.2, and present, respectively, the analysis and simulation results in Sections 3.3–3.4.

[1] As the quality of sensing data decreases with the distance from the target, the CH closer to the target should be activated.

3.1 System Overview

3.1.1 Energy-Based Localization

The fundamental principle applied in the energy-based approaches [46, 73] is that the signal strength (i.e., energy) of a received signal decreases exponentially with the propagation distance [17, 46]:

$$r_i = a \cdot \|x - x_i\|^{-\alpha} + n_i, \qquad 1 \leq i \leq N, \tag{3.1}$$

where r_i is the received signal strength in the i^{th} sensor, $a \in \mathcal{R}$ is the (unknown) strength of an acoustic signal from the target, $x \in \mathcal{R}^2$ is the target position yet to be determined, $x_i \in \mathcal{R}^2$ is the known position of the i^{th} sensor, α is the (known) attenuation coefficient, and n_i is the white Gaussian noise with zero-mean and variance σ^2.

Instead of solving for the unknowns in Eq. (3.1) based on energy readings from multiple sensors, we introduce two simple approaches: one is Voronoi diagram-based, and the other is non-linear optimization based. In the Voronoi diagram-based approach, each pair of energy readings, (r_i, r_j), from two sensors i and j determines a half plane that contains the target, i.e., if $r_i > r_j$, the target is closer to sensor i than to sensor j and hence lies in the half plane that contains sensor i. With multiple pairs of energy readings, the target location can be confined to be the intersection area of all the half planes. It can be shown that for a set of n sensor readings, $n-1$ out of the total $n(n-1)/2$ half planes are independent. Therefore, in order to obtain a bounded intersection area, at least four sensor readings are required. As the locations of sensors are static (only subject to environmental factors such as wind), the intersection area can be determined in advance in the form of Voronoi diagram. For completeness of the thesis, we define the Voronoi diagram below [1]:

Definition 1: (Voronoi) Let $P = \{p_1, \cdots, p_N\}$ be a finite set of points in the n-dimensional space $\mathcal{R}^n$ and their location vectors $l_i \neq l_j, \forall i \neq j$. The region given by $\mathcal{V}(p_i) = \{l \mid \|l - l_i\| \leq \|l - l_j\|, \forall j \neq i\}$ is called *Voronoi cell* associated with p_i and $\mathcal{V}(P) = \bigcup_{i=1}^{N} \mathcal{V}(p_i)$ is said to be the *Voronoi diagram* of P.

We claim that the Voronoi cell $\mathcal{V}(p_i)$ associated with the point p_i is exactly the intersection area of all the half planes by the following observation:

Observation 1: $\mathcal{V}(p_i) = \bigcap_{1 \leq j \leq N, j \neq i} h(p_i, p_j)$, where $h(p_i, p_j)$ is the open half plane containing p_i.

As a result, as long as the sensor with the maximum energy reading, say p_i, can be identified, the location of the target is inside $\mathcal{V}(p_i)$ and the position of p_i can be used as the *approximate* location of the target. Note that the error of the above approximation is small and bounded if the target is at a bounded Voronoi cell $\mathcal{V}(p_i)$. Alternatively, in the case that the CHs have sufficiently high

computation capability, a more sophisticated, nonlinear optimization based localization method can be adopted [46]. Specifically, given a pair of readings, $(r_1, (x_1, y_1))$ and $(r_2, (x_2, y_2))$, where r_1 and r_2 are the signal strength detected at sensor 1 and 2, and (x_1, y_1) and (x_2, y_2) are the locations of sensor 1 and 2, and let $c \stackrel{\triangle}{=} (\frac{r_1}{r_2})^{-2/\alpha} = (\frac{d_1}{d_2})^2$ where d_1 and d_2 are the distances from the estimated target position to sensor 1 and 2, the locus of the potential position of the target, (x, y), can be shown (after a few algebraic operations) to be a circle characterized by

$$(x - x_1)^2 + (y - y_1)^2 = c(x - x_2)^2 + c(y - y_2)^2,$$

or

$$(x - o_x)^2 + (y - o_y)^2 = \rho^2, \tag{3.2}$$

where $(o_x, o_y) = (\frac{cx_2 - x_1}{c-1}, \frac{cy_2 - y_1}{c-1})$ is the center of the circle and

$$\rho = \sqrt{\frac{c}{(c-1)^2} \cdot ((x_1 - x_2)^2 + (y_1 - y_2)^2)}$$

is the radius of the circle.

Suppose the CH receives n replies from its neighboring sensors and one measurement from itself. Then a total of $m = \binom{n+1}{2}$ pairs of readings can be used to estimate the target location, and a nonlinear optimization problem of minimizing the sum of square errors can be formulated as

$$(x, y) = \arg\min_{(x,y)} \sum_{i=1}^{m} \frac{(x - o_{x_i})^2 + (y - o_{y_i})^2 - \rho_i^2}{\rho_i^2}, \tag{3.3}$$

where (o_{x_i}, o_{y_i}) is the center of the circle that gives the locus of the potential location of the target and ρ_i is the radius (Eq. (3.2)). Newton's method [4] can be used to solve the nonlinear programming problem. We will compare the performance of both methods in Section 3.4.

3.1.2 Overview of Proposed Dynamic Clustering Algorithm

As mentioned in Chapter 1, we envision a heterogeneous, hierarchical sensor network that is composed of (a) a static backbone of sparsely placed high-capability sensors called CHs; and (b) moderately to densely populated low-end sensors whose function is to provide sensing information to CHs upon requests. Because of the limited mobility nature of sensors, the calibration process of sensor locations is executed only once when the network is deployed. In this calibration process, geographical information required to construct the Voronoi diagram is collected, and the Voronoi

diagram constructed. Furthermore, both CHs and sensors construct several tables to facilitate their decision on the back-off timer values (to be used when a CH intends to volunteer itself as a leader and when a sensor intends to respond to a CH).

A CH volunteers to become active when it detects that the strength of a received acoustic signal exceeds a pre-determined threshold and the signal matches one of the signal patterns which the system intends to track. As multiple CHs may detect the acoustic signal with a sufficiently high signal-to-noise ratio and volunteer themselves as active leaders, we devise in Section 3.2.2 a two-phase volunteering procedure to determine in a decentralized manner which CH (preferably the one with the strongest signal strength) should be activated.

The tasks of an active CH include the following four steps:

1. broadcasting a packet that contains the energy and the extracted signature[2] of the detected signal to sensors,
2. receiving replies from sensors,
3. estimating the location of the target based on replies,
4. sending the result to subscriber(s).

Upon receipt of a broadcast packet from a CH, a sensor matches the signature with its buffered data. In the case of a match, the sensor then determines, with the use of the Voronoi diagram based table, whether or not it may be (i) the sensor that is closest to the target or (ii) one of the neighbors of the sensor that is closest to the target. If any of the above two conditions holds, it replies, after a random delay, to the CH the strength of the signal it receives. The random delay is determined based on the strength of the signal the sensor detects so as to mitigate collision. We will elaborate on how a sensor determines whether or not, and when, it should respond to a broadcast message from a CH in Section 3.2.3.

Once the CH collects enough replies, it ignores all subsequent replies (if any), generates the localization result and sends the result back to subscribers. Sensors that decided to reply but have not yet done so (as their timers have not expired) do not reply, when they overhear the packet that carries the localization result.

[2]The extracted signature can be either the raw data or the extracted feature of a signal.

Table 3.1: Radio transmission range of Berkeley Motes.

Products	transmission range
MPR300*	30m
MPR400CB	150m
MPR410CB	300m
MPR420CB	300m
MPR500CA	150m
MPR510CA	300m
MPR520CA	300m

$*$ MPR300 is the second generation sensors, while the rest are the third generation sensors.

Table 3.2: Sensing range of several typical sensors

Products	sensing range	typical applications
HMC1002 Magnetometer sensor [95]	5m	Detecting disturbance from automobiles
Reflective type photoelectric sensor [97]	1m	Detecting targets of virtually any material
Thrubeam type photoelectric sensor [97]	10m	Detecting targets of virtually any material
Pyroelectric infrared sensor (RE814S) [96]	30m	Detecting moving objects
Acoustic sensor on Berkeley Motes * [95]	$\sim$ 10m	Detecting acoustic sound sources

* This result is based on our own measurement on Berkeley motes [95].

The relationship between the radio transmission range and the acoustic signal detection range determines the size of a cluster. The radio transmission range is controlled by adjusting the transmission power, while the acoustic signal detection range is controlled by adjusting the detection threshold. If the acoustic signal detection range is larger than the radio transmission range, multiple CHs may become active and multiple clusters are formed at the same time, without knowing the existence of each other. The results obtained from different clusters may differ dramatically because a CH may not be able to recruit sufficient sensors and gather enough sensor information to confine the target in a bounded Voronoi cell. On the other hand, if the radio transmission range is larger than the acoustic signal detection range, the localization results will be more accurate but the collision has to be handled carefully. In the thesis, we assume that the radio transmission range is set to be twice of the acoustic signal detection range. This assumption is corroborated by the field data tabulated in Tables 3.1 and 3.2. In Section 3.4, we will evaluate the performance of the proposed approaches when this assumption holds and does not hold.

3.2 Detailed Description of Proposed Dynamic Clustering Algorithm

The proposed dynamic clustering algorithm is composed of four component mechanisms: initial distance calibration and tabulation, CH volunteering, sensor replying, and reporting of tracking results. In what follows we elaborate on each of the four component mechanisms.

3.2.1 Distance Calibration and Tabulation

Before the acoustic tracking system starts to function, each sensor has to know the positions of sensors in its tracking ranges. Under the assumption that the radio transmission range is larger than the acoustic detection range, a sensor can notify other sensors of its ID, device function (CH or sensor), and location by broadcast. To reduce the possibility of collisions, a broadcast packet is delayed by a back-off value determined based on the sensor ID.

(1) Construction of Voronoi diagrams at a CH After receiving the location information from all the neighboring sensors, a CH constructs two Voronoi diagrams around itself, one for the set of neighboring sensors and the other for the set of neighboring CHs. In what follows, we first elaborate on how to use Voronoi diagrams to construct response tables and will explain their usage in Sections 3.2.2–3.2.3.

(2) Construction of the response table based on Voronoi diagrams at a CH After the Voronoi diagrams are constructed, a CH proceeds to construct the *response table* to facilitate its determination of back-off timer values (to be used when the CH intends to volunteer itself as a leader). The table is indexed by the estimated distance from the target to the CH, d, and each table entry stores the conditional probability that with the distance d, a target indeed locates within this CH's Voronoi cell.

The distance from the target to a CH, say CH_i, can be estimated (with the noise in Eq. (3.1) ignored [3] and the conjectured signal strength from the target) as $d = (r/a)^{-1/\alpha}$, where r is the received signal strength. That is, the possible position at which the target may be located is a circle centered at CH_i and with radius d. Initially the signal strength, a, of the target is set to a default

[3] The noise has little effect on the estimate of the probability values in the response table because (i) the noise is assumed to be Gaussian with zero mean; (ii) the estimate of probability values is based on the average of all the samples on the circle. In addition, the effect of the noise on the overall performance will be investigated in Section 3.4 (e.g., Fig. 3.8).

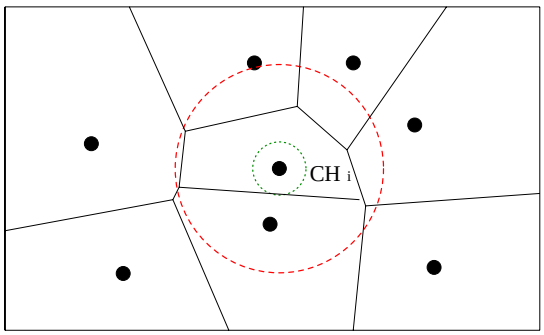


Figure 3.1: Voronoi diagram of CH_i. If a target locates within the inner dotted circle, CH_i is sure that the target is in its Voronoi cell. On the other hand, if a target resides outside the outer dashed circle, CH_i is sure that the target is out of its Voronoi cell.

value and dynamically adjusted according to the current localization result and the collected data. Specifically, after locating the target position, an active CH has n pairs of readings (r_i, d_i) from n sensors and one pair of readings from itself, where r_i is the energy reading replied from sensor i and d_i is the distance from the estimated target position to sensor i. The CH then estimates the signal strength of the target based on these $n+1$ pairs of (r_i, d_i) as follows:

$$a = (\sum_{i=1}^{n+1} \frac{r_i}{d_i^{-\alpha}})/(n+1). \tag{3.4}$$

Next, we derive the conditional probability, $\Pr(i|d)$, that the target locates within the Voronoi cell of CH_i, given the distance from the target to CH_i, d. Let $d_{i,min}$ and $d_{i,max}$ denote, respectively, the distance from CH_i to its nearest neighboring CH and that from CH_i to the farthest Voronoi vertex of its Voronoi cell. We have to consider three cases (Figure. 3.1)

(i) $d < 1/2 \cdot d_{i,min}$: CH_i is the nearest CH to the target because the circle that is centered at CH_i and has a radius of d lies completely within CH_i's Voronoi cell. Hence $\Pr(i|d) = 1.0$.

(ii) $d > d_{i,max}$: By the definition of $d_{i,max}$, the circle centered at CH_i with a radius of d lies completely outside CH_i's Voronoi cell. Hence $\Pr(i|d) = 0.0$.

(iii) $1/2 \cdot d_{i,min} < d < d_{i,max}$: The circle that is centered at CH_i and has a radius of d is partially located within CH_i's Voronoi cell and hence $0.0 \leq \Pr(i|d) \leq 1.0$. We will further estimate $\Pr(i|d)$ using the algorithm given in Fig. 3.2.

Note that lines 5 and 7 in the algorithm can be executed in $O(logn)$ time and $O(n)$ space, where n is the number of CHs in the detection range, after the Voronoi diagram is constructed. Essentially the

```
CALC Pr(i|d)(d)
1.  gain ← 0; loss ← 0
2.  for j ← 1 to resolution
3.      θ ← 2π · j/resolution
4.      x ← CH_i.x + d · cos(θ); y ← CH_i.y + d · sin(θ)
5.      if (x, y) is within Voronoi cell V(CH_i)
6.        gain++
7.      else if (x, y) is within V(CH_k) and dist((x, y), CH_k) > 1/2 · d_{k,min}
8.        loss++
9.  return gain/(gain+loss)
```

Figure 3.2: The algorithm that calculates $\Pr(i|d)$ when $1/2 \cdot d_{i,min} < d < d_{i,max}$.

algorithm takes $(360/resolution)$ samples on the circle that is centered at CH_i and has a radius of d, where *resolution* is a tunable parameter. Whenever a sample lies within $\mathcal{V}(CH_i)$, the variable *gain* is incremented. On the other hand, when a sample may lie within $\mathcal{V}(CH_k)$ for some neighboring CH_k, the variable *loss* is incremented, except that samples that are surely located in the Voronoi cell of one neighboring CH are not counted in *loss*. The probability sought for $\Pr(i|d)$ is then estimated as $\frac{gain}{gain+loss}$.

Note also that in line 7 CH_i needs to know $d_{k,min}$ for every neighboring CH_k. This can be achieved by having each CH broadcast the distance to its neighboring CHs in the second round. Also, if CH_i's Voronoi cell is not bounded, the algorithm excludes in line 5 samples which are in $\mathcal{V}(CH_i)$ but with the distance to CH_i larger than that from CH_i to its farthest Voronoi vertex.

To keep the table size at a fixed value, we quantize the distance from the target to itself d as follows. The table contains k entries. The first entry is indexed by the maximum value of d (denoted as d_{min}) such that $\Pr(i|d_{min}) = 1.0$, while the last entry by the minimum value of d (denoted as d_{max}) such that $\Pr(i|d_{max}) = 0.0$. The other $k-2$ entries are indexed by values evenly spaced between d_{min} and d_{max}. Given an arbitrary distance d, a binary search is made to locate the two entries with the closest distance and interpolation is used to obtain the approximate value of $\Pr(i|d)$.

(3) Construction of the Voronoi diagram and the response tables at a sensor A sensor S_j constructs one Voronoi diagram around itself for the set of neighboring sensors. The response table is indexed by the ratio, $r_{i \to j}$, of the signal strength at a CH_i to that at itself, where CH_i is the CH that initiates the solicitation. Each table entry stores the conditional probability, $\Pr(j|r_{i \to j})$, that the target locates in S_j's Voronoi cell $\mathcal{V}(S_j)$ rather than other neighboring sensors or CHs,

given the ratio $r_{i\to j}$. Specifically, suppose a sensor S_j, located at (x_2, y_2), receives an information solicitation packet from a CH CH_i, located at (x_1, y_1). Assume that the signal strength detected at CH_i and S_j, is, respectively, r_1 and r_2. Then $r_{i\to j} = \frac{r_1}{r_2} = (\frac{d_1}{d_2})^{-\alpha}$. Recall that $c \triangleq (\frac{r_1}{r_2})^{-2/\alpha} = (\frac{d_1}{d_2})^2$. After a few algebraic operations, one can derive that the locus of the potential position of the target, (x, y), is a circle characterized by Eq. (3.2)

$$(x - o_x)^2 + (y - o_y)^2 = \rho^2,$$

where $(o_x, o_y) = (\frac{cx_2 - x_1}{c-1}, \frac{cy_2 - y_1}{c-1})$ is the center of the circle and

$$\rho = \sqrt{\frac{c}{(c-1)^2} \cdot ((x_1 - x_2)^2 + (y_1 - y_2)^2)}$$

is the radius of the circle.

Given the above expressions and the signal strength detected at CH_i, each sensor can locate the potential positions of the target. Using a similar algorithm to that in Fig. 3.2, we can calculate and tabulate $\Pr(j|r_{i\to j})$. Each sensor S_j then maintains, for each CH within its transmission range, a table of k entries. In addition, let $\overline{N_j}$ denotes the event that the target is in neither $\mathcal{V}(S_j)$ nor the Voronoi cells of any of S_j's Voronoi neighbors. Then each sensor calculates the conditional probability, $\Pr(\overline{N_j}|r_{i\to j})$, that the target is located at neither $\mathcal{V}(S_j)$ nor the Voronoi cells of any of S_j's Voronoi neighbors. Note that S_j needs only to store the minimum value of $r_{i\to j}$, r_{min}, such that $\Pr(\overline{N_j}|r_{min}) = 1.0$.

One point is worthy of mentioning — the process of table construction takes place only once at each CH/sensor at the time when the sensor network is deployed. In the case that sensors are not equipped with sufficient circuitry to perform this pre-processing, their nearest CH may construct the tables on their behalves and transmit the resulting tables to them.

3.2.2 Cluster Head Volunteering

In the first step of dynamic clustering, a CH volunteers to recruit sensors to form a cluster, if its detected signal strength exceeds a predefined threshold. One fundamental design issue to consider is which CH(s) should be elected to form a cluster if more than one CHs detect simultaneously signals with strength exceeding the threshold. Note that if two or more clusters are active simultaneously, packets exchanged in one cluster may interfere/collide with those in the other cluster(s). (This is

corroborated by our performance study in Sec. 3.3 in which we will investigate the performance in the case that more than one cluster is active.) Ideally, the CH that is closest to the target (or the CH with the largest SNR ratio) should be elected. Since the communication cost of deterministic leader election is very high, we propose to use a two-phase, random delay-based broadcast mechanism to implicitly determine the active CH. In what follows, we first describe how the random delay is set and then delve into the two-phase broadcast mechanism.

Determination of back-off timer values Without a centralized facility and/or excessive message exchanges for CH election, an effective method to determine the active CH is to figure in the received signal strength into the determination of the back-off timer values used to send solicitation packets. A CH whose received signal strength exceeds the pre-determined threshold sets a back-off timer and does not broadcast its solicitation packet until the timer expires. If by the time the back-off timer expires, the CH receives a solicitation packet from some other CH, it cancels the timer. Specifically, the back-off time, D, for which CH_i (with an estimated distance, d, from the target to itself) uses is:

$$D = W_{min} + (W_{max} - W_{min}) \cdot (1 - \Pr(i|d)) + U(W_{ran}), \tag{3.5}$$

where W_{min} and W_{max} denote the minimum and maximum backoff timer values, $U()$ is the uniform distribution in $[0, W_{ran} - 1]$, and $\Pr(i|d)$ can be retrieved from the response table (Section 3.2.1). Note that D contains two parts. The first two terms in Eq. (3.5) are the deterministic part that relates the estimated distance to the back-off delay value, and the third term accounts for the random part that prevents potential collision when the distances from the target to two or more CHs are approximately the same. The random part is an order of magnitude smaller than the deterministic part. Note also that the back-off timer is set at the application level, meaning that a CH only passes an information solicitation broadcast packet onto its MAC layer when the back-off timer expires. An underlying carrier-sense MAC protocol, such as a light-weight version of IEEE 802.11b, is still needed to mitigate collision at the MAC level.

Two-phase broadcast with energy and signature packets Although the above random back-off delay method reduces the possibility of collision, it does not totally eliminate it. This is in part due to the fact that these information solicitation packets may be queued at the MAC layer

before being transmitted. Moreover, in order for sensors to identify which event a CH is interested in, information solicitation packets contain not only the signal strength but also the signature of the acoustic sound [4] and as a result are usually quite large. A sensor then attempts to match the received signature with its buffered data to make sure the same event is being tracked. These solicitation packets of large sizes also contribute to the likelihood of collision.

To deal with the above problems, we propose a two-phase broadcast mechanism: in the first phase, an *energy packet* that carries only the signal strength information is broadcast. In the second phase, a *signature packet* that contains the detailed signature information is then broadcast. Both packets are subject to the same random back-off delay value (except for the randomized term). Specifically the two-phase broadcast mechanism operates as follows:

(R1) A CH sets its back-off timer with the value of D (Eq. (3.5)) for the energy packet.

(R2) When the timer in the first phase expires, the CH broadcasts an energy packet and sets its back-off timer to the same value in the second phase (except for the randomized term). If by the time the timer in either the first or second phase expires, the CH overhears a broadcast packet with stronger signal strength or a significant packet, it cancels its timer and does not henceforth participate in the volunteering process. Otherwise, the overheard packet is ignored.

(R3) When the timer in the second phase expires, the CH broadcasts a signature packet.

Note that the energy packet in the first phase is much shorter than the signature packet in the second phase and hence the possibility of collision is reduced. Also, even if the energy packet of a CH collides with other packets, the CH still schedules the broadcast of its signature packet in the second phase, unless it overhears another signature packet by the time its second-phase timer expires.

Operations performed by a volunteering CH In order to realize the above mechanism, clocks have to be synchronized for two reasons: first, a CH has to inform sensors of the time when the event takes place so that sensors can match the signature contained in the packet with its buffered data. Second, when a CH sends the localization result to the sink, it has to inform the

[4] The signature information increases system robustness when one deals with single or multiple targets.

sink of the time when the event takes place. To avoid the expensive clock synchronization process, we use relative clock values rather than absolute clock values. A CH records the time it detects the event. Before broadcasting the signature packet, the CH calculates the time lag (which includes the processing time, back-off delay and expected propagation delay), and attaches it in the signature packet. When a sensor receives the signature packet, it can infer the time when the CH detected the event.

After broadcasting the signature packet, a CH sets a timer to wait for replies from neighboring sensors. If a sufficient number of replies are received before the timer expires, the CH cancels the timer and calculates the localization result; otherwise, the result is generated upon the expiry of the timer.

3.2.3 Sensor Replying

After a sensor receives a signature packet, it attempts to match the signature with its buffered data. The search range can be confined with the use of the time lag information contained in the signature packet. If the signature is matched, the signal strength in the buffered data (as well as the ratio of the signal strength detected at the CH to that in the buffered data) is calculated.

A sensor S_j does not respond if $\Pr(j|r_{i\to j}) = 0$ and $r_{i\to j} \geq c' \cdot r_{min}$, where c' is a constant and r_{min} is the minimum value of $r_{i\to j}$ such that $\Pr(\overline{N_j}|r_{min}) = 1.0$. In the case that a response is to be sent, a similar, random back-off method is used to avoid potential collision of all the replies from sensors to CH_i. That is, a sensor S_j delays its reply by a back-off value determined by

$$D' = W'_{min} + (W'_{max} - W'_{min}) \cdot (1 - \Pr(j|r_{i\to j})) + U(W'_{ran}),$$

where W'_{min} and W'_{max} are the minimum and maximum back-off values, and $\Pr(j|r_{i\to j})$ can be retrieved from the response table (Section 3.2.1).

If by the time the back-off timer expires the sensor overhears reply packets from other sensors, it records the sensor that reports the largest signal strength. When the timer expires, the sensor sends a reply packet only if (i) the signal strength it detected is larger than that carried in any of the overheard reply packets; or (ii) it is one of the Voronoi neighbors of the sensor node that reports the largest signal strength. The two conditions ensure that the set of replies that the active CH will receive includes a reply from a node S_j with the largest signal strength, and replies from all

S_j's Voronoi neighbors. Due to the convexity of the propagation model, the target is guaranteed to be within the Voronoi cell of S_j, $\mathcal{V}(S_j)$. Finally, when a sensor overhears a tracking report packet from a CH, if it does prepare to reply to this CH for the same event, the sensor cancels its timer and does not respond.

3.2.4 Reporting Tracking Results

A CH generates the localization result, if either the timer set after the signature packet was sent expires or the CH receives a sufficient number of replies, whichever occurs first. By "sufficient," we mean the set of replies includes a reply from a node S_j with the largest signal strength, and replies from all S_j's Voronoi neighbors. In this manner, the target is guaranteed to be within the Voronoi cell of S_j, $\mathcal{V}(S_j)$, due to the convexity of the propagation model. Then, the CH may use a localization method to determine the position of the target. As discussed in Section 3.1, there are two simple localization approaches: one is Voronoi diagram based and the other is nonlinear optimization based. In the Voronoi diagram based approach, the position of the sensor that reports the largest signal strength is used as the estimate of the target position. Although the estimate may be inaccurate, this approach is usually robust and the estimation error is bounded. The nonlinear optimization based approach, on the other hand, attempts to find the location (x, y) by minimizing the sum of square errors from $(n + 1)$ pairs of readings [46] [5]. As one can expect, the second localization method achieves better estimation results at the expense of higher computational costs.

Once the localization result is generated, the CH has to send the location and time information of the target to the sink(s). The CH calculates the time difference between the time instant when it detected the event and the expected time instant when the packet arrives at the next hop and sends the tracking report packet. Each intermediate router accumulates and updates the time lag in the same manner. When the sink receives the report, it can calculate the time when the CH detected the event.

3.3 Analysis of Proposed Algorithm

In this section we analyze the performance of the CH volunteering procedure (Section 3.2.2) and show that with properly selected parameters W_{min}, W_{max}, and W_{ran}, the following properties hold:

[5] Eq. (3.3) differs from that of [46] in that each estimation error term is normalized by the square of radius, ρ_i^2.

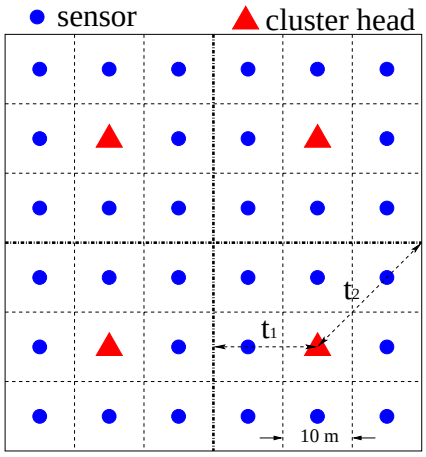


Figure 3.3: The scenario used in the analysis.

(a) the CH with the largest signal strength (CH_1) transmits its energy packet earlier than any other CH (and thus suppress other energy packets) with a large probability; and (b) in the case that (a) is not true, CH_1 may still become the leader eventually. To facilitate derivation, we make the following simplifying assumptions:

1. Both the CHs and the sensors are deployed on a regular square grid shown in Fig. 3.3. In this scenario, at most four CHs will contend to become a leader.

2. Instead of Eq. (3.5), the following equation is used by CH_i to determine its back-off time:

$$D = W_{min} + (W_{max} - W_{min}) \cdot (\frac{d - t_1}{t_2 - t_1}) + U(W_{ran}), \tag{3.6}$$

   where d is the estimated distance from the target to CH_i, t_1 is half of the distance from CH_i to the closest neighboring CH, and t_2 is the distance from CH_i to its farthest Voronoi vertex. In the scenario given in Fig. 3.3, $t_2 = \sqrt{2} \cdot t_1$. Note that we replace $1 - \Pr(i|d)$ in Eq. (3.5) with $\frac{d-t_1}{t_2-t_1}$ to simplify the analysis.

Given the position of a target, we number CHs in the increasing order of their distances to the target, such that if d_i denotes the distance from CH_i to the target, we have $d_1 < d_2 < \cdots < d_N$. Ideally by the end of the volunteering process, CH_1 should become the leader. We consider three cases in the proposed two-phase broadcast mechanism:

- Case (i): CH_1 transmits its energy packet earlier than any other CHs. Also, by overhearing CH_1's energy packet, other CHs cancel their back-off timers and do not volunteer.

- Case (ii): CH_i, for some $i > 1$ successfully transmits its energy packet earlier than CH_1.[6] By (R2) in Section 3.2.2, CH_1 proceeds to transmit its energy packet, and meanwhile this packet does not collide with energy packets from other CHs. Under this condition, if CH_1's energy packet is sent earlier than CH_i's signature packet, CH_1 will still becomes the leader. We will show in Lemma 2 that with proper parameter setting, CH_1's energy packet will always be sent earlier than CH_i's signature packet, and hence CH_1 will become the leader.

- Case (iii): The energy packet sent by CH_1 collides with that by CH_j, for some $j > 1$. In this case, CH_1 may still become active if its signature packet is sent earlier than any other signature packets and does not collide with other packets.

First we show in Lemma 1 that if the parameters W_{min}, W_{max}, and W_{ran} are properly selected, cases (ii) and (iii) occur with small probabilities.

Lemma 1: Cases (ii) and (iii) occur only if $d_i - d_1 \leq \delta$, where $\delta = \frac{W_{ran}-1}{W_{max}-W_{min}} \cdot (t_2 - t_1)$, where W_{min} and W_{max} are the maximum and minimum backoff timer values and $[0, W_{ran} - 1]$ is the range of the uniform distribution in Eq. (3.6).

Proof: Cases (ii) and (iii) occur only if $D_i \leq D_1$, for some i, where D_i is the backoff time determined by Eq. (3.6). That is,

$$D_i = W_{min} + (W_{max} - W_{min}) \cdot (\frac{d_i - t_1}{t_2 - t_1}) + U(W_{ran})$$

$$\leq W_{min} + (W_{max} - W_{min}) \cdot (\frac{d_1 - t_1}{t_2 - t_1}) + U(W_{ran}) = D_1.$$

As $U(W_{ran})$ is a discrete random variable with the uniform distribution in $[0, W_{ran} - 1]$, we set $U(W_{ran})$ in D_i to be zero and $U(W_{ran})$ in D_1 to be $W_{ran} - 1$ for the worse case. Then we can reach the following conclusion

$$d_i - d_1 \leq \frac{W_{ran} - 1}{W_{max} - W_{min}} \cdot (t_2 - t_1) \triangleq \delta.$$

■

In the thesis we set $W_{ran} \ll W_{max}$ (approximately three orders of magnitude smaller) to make δ a very small value, thus ensuring CH_1 transmits its energy packet earlier than any other packets with a high probability.

Lemma 2: In case (ii), CH_1 transmits its energy packet earlier than the signature packet from CH_i if $W_{ran} < W_{min} + (W_{max} - W_{min}) \cdot (\frac{2d_i - d_1 - t_1}{t_2 - t_1}) + 1$.

Proof: As discussed in Section 3.2.2, after CH_i sends its energy packet, it has to wait for D_i' units of time before sending the signature packet. Moreover, if CH_i overhears an energy packet with larger signal strength before its second-phase timer expires, it cancels the timer and does not send the signature packet. Therefore, as long as CH_1 sends its energy packet before $D_i + D_i'$, CH_i will not send the signature packet. That is,

$$D_1 < D_i + D_i'.$$

[6] There may be multiple such CHs.

After a few algebraic operations, we have

$$W_{ran} < W_{min} + (W_{max} - W_{min}) \cdot (\frac{2d_i - d_1 - t_1}{t_2 - t_1}) + 1.$$

■

In the thesis we set W_{ran} to be equal to W_{min} such that the inequality in Lemma 2 holds, thus ensuring in case (ii) CH_1 always becomes the leader.

Now we inspect the probability that CH_1 will not become a leader in case (iii). In this case, CH_1 may still become the leader if (a) CH_j's signature packet is not scheduled before CH_1's signature packet and (b) CH_1's signature packet does not collide with any other signature or energy packets. As the probability of the above events can not be exactly and easily calculated, we use the probability that CH_1's energy packet collides with other energy packets as the *upper* bound of the probability that CH_1 can not become the leader. We will show that this probability is very small.

The first step to calculate the probability that CH_1's energy packet collides with other energy packets is to identify the region in the Voronoi diagram in which collisions may potentially occur. By Lemma 1, any point in such a region possesses the following property: The difference between the distance from this point to CH_i, d_i, and that to CH_1, d_1, is less than δ. As $d_i - d_1 = \delta$ is a hyperbola in the Voronoi diagram, the region is bounded by the hyperbolas and the Voronoi cell boundaries. For example, if the CHs and sensors are deployed in the square grid in Fig. 3.3, then the hyperbolas that bound the collision region are given in Fig. 3.4. Because CH_1's Voronoi cell is symmetric, we only consider its one eighth portion in the northwestern corner (i.e., the area bounded by CH_1, P_0, and P_1).

The collision region under consideration is composed of four sub-regions, $\mathcal{R}_1(CH_1, P_2, P_3)$, $\mathcal{R}_2(P_1, P_4, P_3, P_2)$, $\mathcal{R}_3(P_3, P_4, P_5)$, and $\mathcal{R}_4(P_4, P_0, P_5)$. (Points in the parenthesis represent the vertices of the sub-regions in the clockwise direction.) If the target is located in region $\mathcal{R}_1$, by Lemma 1 CH_1 sends its energy packet earlier than any other CHs. On the other hand, if the target is located in region $\mathcal{R}_i, i = 2, 3, 4$, CH_1's energy packet may collide with $(i-1)$ other energy packets. For example, if the target is located in region $\mathcal{R}_3$, CH_1's energy packet may potentially collide with energy packets sent by the CHs in the left lower grid and in the right upper grid. Given that the target is located at $p = (x, y)$ in region $\mathcal{R}_i, i = 2, 3, 4$, the conditional probabil-

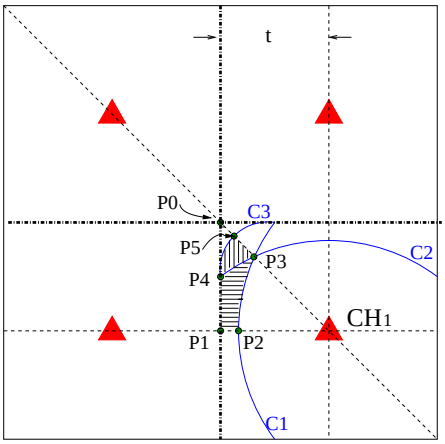


Figure 3.4: $\mathcal{R}_1(CH_1, P_2, P_3)$, $\mathcal{R}_2(P_1, P_4, P_3, P_2)$, $\mathcal{R}_3(P_3, P_4, P_5)$, and $\mathcal{R}_4(P_4, P_0, P_5)$. If the target is located in region $\mathcal{R}_1$, CH_1 sends its energy packet earlier than any other CHs. If the target is located in region $\mathcal{R}_i, i = 2, 3, 4$, CH_1's energy packet may collide with $(i-1)$ other energy packets. $\overline{P_1P_2} = \delta/2$.

Region	$Pr[CH_1\ collision \mid p = (x, y)]$
$\mathcal{R}_2$	$\sum_{t=W_2}^{W_1+W_{ran}-1} \frac{1}{W_{ran}^2} = \frac{W_{ran}+W_1-W_2}{W_{ran}^2}$
$\mathcal{R}_3$	$\sum_{t=W_2}^{W_3-1} \frac{1}{W_{ran}^2} + \sum_{t=W_3}^{W_1+W_{ran}-1} \{\frac{1}{W_{ran}^3} + \binom{2}{1}\frac{1}{W_{ran}^2}\} = \frac{W_3-W_2}{W_{ran}^2} + (\frac{1+2W_{ran}}{W_{ran}^3})(W_{ran}+W_1-W_3)$
$\mathcal{R}_4$	$\sum_{t=W_2}^{W_3-1} \frac{1}{W_{ran}^2} + \sum_{t=W_3}^{W_4-1} \{\frac{1}{W_{ran}^3} + \binom{2}{1}\frac{1}{W_{ran}^2}\} + \sum_{t=W_4}^{W_1+W_{ran}-1} \{\frac{1}{W_{ran}^4} + \binom{3}{2}\frac{1}{W_{ran}^3} + \binom{3}{1}\frac{1}{W_{ran}^2}\} = \frac{W_3-W_2}{W_{ran}^2} + \frac{1+2W_{ran}}{W_{ran}^3}(W_4-W_3) + \frac{1+3W_{ran}+3W_{ran}^2}{W_{ran}^4}(W_{ran}+W_1-W_4)$

Table 3.3: Conditional probability that CH_1's energy packet collides with other energy packets when the target is located at $\mathcal{R}_i$. W_i is the deterministic part of the backoff timer value of CH_i.

ity, Pr(collision of CH_1's energy packet$|p = (x, y)$), that CH_1's energy packet collides with other energy packets is given in Table 3.3.

We illustrate how to derive this conditional probability when the target in region $\mathcal{R}_4$. The relationship of the values of CH_i's backoff timers $(1 \le i \le 4)$ is depicted in Fig. 3.5. Recall that each timer value contains a deterministic part (W_i) and a randomized part (uniform in $[0, W_{ran}]$). Since $d_1 < d_2 < d_3 < d_4$, by Eq. (3.6) we have $W_1 < W_2 < W_3 < W_4$. The conditional probability is calculated by considering the conditions in different segments of the backoff period. From time W_2 to $\min\{W_3 - 1, W_1 + W_{ran} - 1\}$, only CH_1 and CH_2 may contend with each other. If both CH_1's and CH_2's timers expire in the same time slot (with probability $\frac{1}{W_{ran}^2}$), the collision occurs. Thus, the first term in the conditional probability of $\mathcal{R}_4$ in Table 3.3 is the probability that CH_1 CH_2 collide in $[W_2, W_3 - 1]$. Derivation of the other terms can be similarly explained.

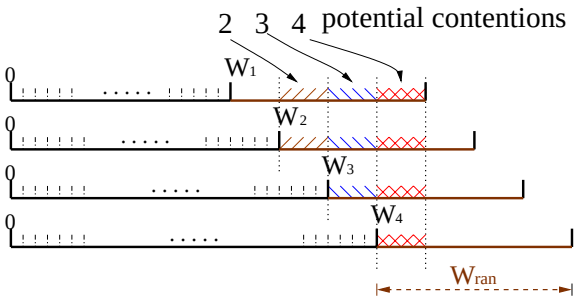


Figure 3.5: An example that shows how the backoff timer of CH_1 may overlap that of CH_i, $i = 2, 3, 4$, when the target is in region $\mathcal{R}_4$. W_i is the deterministic part of CH_i's backoff timer value, and W_{ran} is the randomized part (that is uniformly distributed in $[0, W_{ran}]$.

Finally the probability that CH_1's energy packet collides with other energy packets can be expressed as:

$$\Pr(\text{collision of } CH_1\text{'s energy pkt}) = \frac{2}{t^2} \cdot \sum_{i=2}^{4} \{ \int_{\mathcal{R}_i} \Pr(\text{collision of } CH_1\text{'s energy pkt} | p = (x, y)) \cdot dxdy \}.$$

In the thesis we set W_{max}, W_{min}, W_{ran} to be 0.1 second, 10^{-4} second, and 10^{-4} second, respectively.[7] In this case $\delta = 3.317 \times 10^{-4} t$, region $\mathcal{R}_i$ is much smaller than region $\mathcal{R}_1$, and $\Pr(CH_1$'s energy pkt collides with other energy pkts) is equal to 5.3×10^{-5}. This ensures the probability that CH_1's energy packet collides with other energy packets is almost zero, and the proposed algorithm elects with high probability the CH with the largest signal strength to be the leader.

3.4 Simulation Results

In this section, we evaluate the proposed dynamic clustering algorithm using *SensorSim* [99] (which is built upon *ns-2*[98]). The surveillance area is 180×180 m^2. A total of 324 devices, including 36 CHs and 288 sensors, are deployed in the area. In addition, a sink is located at (0,0) in the system. Two deployment configurations are considered: in one configuration the CHs and sensors are deployed on a square grid shown in Fig. 3.3, and in the other both the CHs and sensors are uniformly distributed in the area. A static topology based sink tree rooted at the sink is built by the underlying routing protocol and IEEE 802.11b is adopted as the underlying MAC protocol. The detection threshold of each acoustic microphone is adjusted so that the acoustic detection range is 25 meters. The radio transmission range is set to be twice of the acoustic detecting range, i.e., 50 meters. The magnitude of the signal from the target, a, stays constant during the tracking period.

[7] If the slot time is 20 μ second, W_{ran} can be divided into 5 slots.

We will relax the above two settings later and investigate the effects of various acoustic detection ranges and varying signal strength of the target on the performance.

The performance metrics of interest are (i) location error: the deviation (in meters) of the estimated location from the exact location of the target; (ii) latency: the time interval from the instant when the acoustic event occurs till the time instant when the location result is delivered to the sink; (iii) the number of events detected and reported to the sink; (iv) the number of collisions throughout the simulation; and (v) the total number of control messages (information solicitation messages, replies, and tracking reports to the sink) throughout the simulation. Each data point reported below is an average of 30 simulation runs.

In the first set of simulations, we compare the performance of the full-fledged version of the proposed algorithm, including the distance calibration and tabulation procedure and the two phase CH volunteering procedure, against two partial versions of the proposed algorithm and the static clustering approach. The first partial version includes only one phase in CH volunteering, i.e., each time a CH intends to volunteer itself as the leader, it sends a complete signature packet after its timer expires. Also, the first version does not construct the tables in advance to facilitate determination of the back-off timer values. Instead a CH determines its backoff timer value based on Eq. (3.6), and a sensor sets its backoff timer in proportion to the ratio of the signal strength detected by the soliciting CH to that detected by itself. The second partial version employs the two phase CH volunteering procedure but does not exercise the distance calibration and tabulation procedure. The Voronoi diagram based localization approach is used to estimate the target position. In the simulation, the target generates an acoustic event every 0.5 second. The target moves continuously at the speed of maximum 20 m/s under the random waypoint model. As each simulation run lasts for 1000 seconds, at most 2000 events can be detected.

Table 3.4 gives the comparison results under the first configuration (square deployment). Under the assumption of negligible noises, each sensor can precisely estimate the distance from the target to itself based on the magnitude of its received signal strength. The proposed approach incurs the minimum location error, the smallest latency, and the least amount of message exchanges. The static clustering approach incurs much larger location error, as a CH that is not in the vicinity of the target may be responsible for estimating the location and reporting the event. Without the two-phase volunteering procedure, the first partial version of the proposed algorithm incurs significant collision. This is in part because once a solicitation packet is handed over to the MAC layer for transmission, it cannot be canceled by the proposed algorithm even if the CH overhears a packet

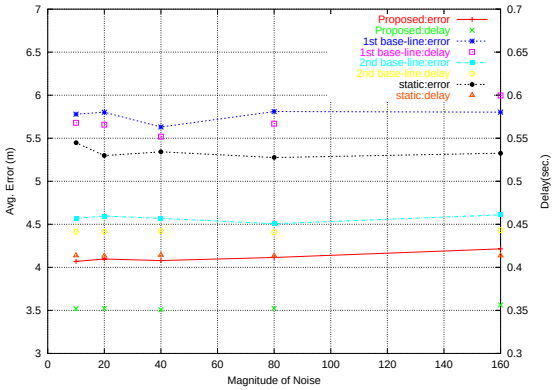
	avg err(m)	latency(s)	detected events	collision times	tot msg sent
static cluster	5.57	0.42	1926	8574	29652
1st base-line	5.89	0.57	1980	30634	57505
2nd base-line	4.87	0.45	1998	10664	40824
proposed algorithm	4.35	0.33	1989	464	17513

Table 3.4: Comparison between the full-fledged version of the proposed algorithm, two partial versions of the proposed algorithm, and the static clustering algorithm under square deployment and the assumption of negligible noises.

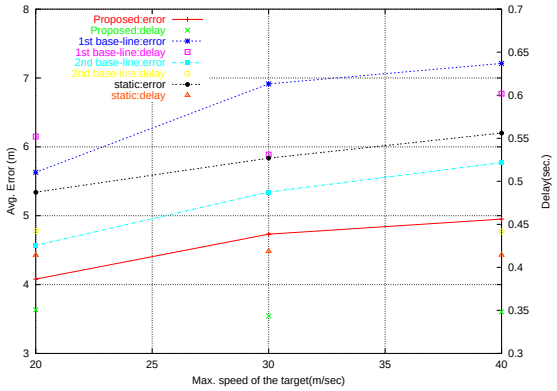
with larger signal strength. In the case that the solicitation packet contains both the signal strength and the signature, the chance for MAC level collision increases. This problem is mitigated in the second partial version. However, without carefully setting the backoff timer values, the second partial version still incurs significant message overhead. In contrast, the full-fledged version incurs the least collision and the least amount of message exchanges, the former due to the two-phase CH volunteering procedure, and the latter due to the use of Voronoi diagram to properly set up the backoff timer values.

Effects of noise and moving speed of the target under the scenario of square deployment In the next set of simulations, we investigate the impact of noise on the performance of the proposed algorithm. The noise for each sensor, n_i in Eq. (3.1), is the product of noise magnitude and additive Gaussian distribution $\mathcal{N}(0, 1.0)$. Fig. 3.6 (a) gives the average location error and latency under different magnitudes of noise. The noise with magnitude 40 is approximately equivalent to SNR = 10. The proposed algorithm performs much better than the static clustering algorithm. This is partially because the proposed algorithm has figured in the effect of possible errors on measurements and can tolerate certain levels of noise. Fig. 3.6 (b) depicts the effect of the moving speed of the target on the performance (with the noise magnitude fixed at 40). Even though the performance of the proposed algorithm degrades as the moving speed of the target increases, it is still the most robust algorithm as compared to other three approaches.

Effects of noise and moving speed of the target under the scenario of random deployment Here we investigate the performance of the proposed algorithm under the assumption that both CHs and sensors are uniformly distributed in the area. Fig. 3.7(a) gives the results of the full-fledged and partial versions of the proposed algorithm and the static cluster approach under nine deployment configurations with noise = 40 and maximum speed = 20 (m/s). The full-fledged version of the proposed approach outperforms two other partial versions and the static clustering approach, although its performance also degrades. This is due to the fact that as CHs and sen-



(a) Performance under different magnitudes of noise

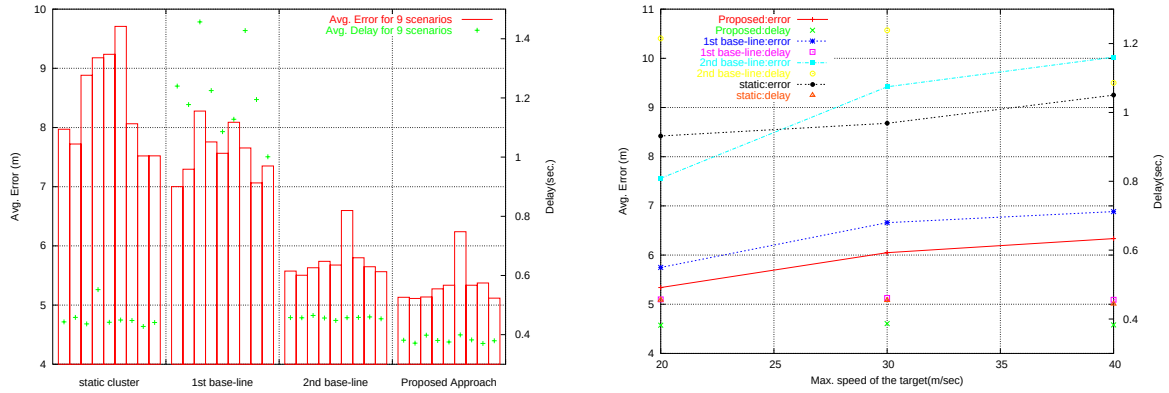


(b) Performance under different moving speeds of the target

Figure 3.6: Performance of the proposed algorithm under different magnitudes of noise ((a)) and different moving speeds of the target ((b)) in the scenario of square deployment.

sors are not deterministically deployed, the average distance from the target to its closest sensor also becomes larger. Another interesting observation is that the gap between the first and second partial versions is significant. This demonstrates the usefulness of the two phase CH volunteering procedure. Fig. 3.7(b) gives the average performance under the nine deployment configurations as the moving speed of the target varies. The result exhibits similar trends to that under square deployment.

Comparison of two localization methods Next we compare the performance of two different localization methods introduced in Section 3.1.1 under the scenario of random deployment. In the first Voronoi diagram based localization method, the position of the sensor that reports the largest signal strength is taken as the estimate of the target position, while in the second localization approach, the target location (x, y) is searched by solving a nonlinear optimization given in Eq. (3.3) to minimize the sum of normalized estimation errors. Fig. 3.8 gives the results of different combinations of clustering and localization approaches with respect to different values of noise magnitudes. As expected, the nonlinear optimization based localization method incurs much less estimation error as compared to the Voronoi diagram based method. With a more accurate localization method, the performance gap between the proposed approach and the static clustering approach becomes larger. This is because in the proposed approach an active CH is able to collect readings from sensors closer to the target and thus the estimate is more robust to noises. Another interesting finding in Fig. 3.8 is that the performance of the nonlinear optimization based localization method degrades more rapidly as the magnitude of noise increases. This is perhaps attributed to the fact



(a) (b)

Figure 3.7: Performance of the full-fledged and partial versions of the proposed algorithm under nine different random deployment configurations with noise = 40 and maximum speed = 20 ((a)) and with noise =40 and varying moving speeds.

that the Voronoi diagram based localization method relies only on one maximum reading. Under a very low SNR, the probability that a sensor which is far away from the target detects the maximum signal strength is small.

Effect of different acoustic detection ranges As discussed in Section 3.1.2, we assume that the radio transmission range is at least twice as large as the acoustic detection range to avoid the case in which more than one clusters form. Although the assumption is corroborated by the field data given in Tables 3.1 and 3.2, we nevertheless investigate the effect of varying acoustic detection ranges on the performance. We fix the radio transmission power and the thresholds of detecting the radio and acoustic signals, but vary the magnitudes of the acoustic source so that the ratio of acoustic detection range to the radio transmission range changes accordingly. Fig. 3.9 gives the performance (with the noise magnitude fixed at 40 and the maximum speed equal to 20 m/s) of the proposed approach and the the static clustering approach with respect different ratios. When the ratio is very small, the estimation error is small but CHs fail to detect most acoustic events because of the insufficient coverage. More sensors and CHs are needed to be deployed to detect targets of small acoustic magnitudes. On the other hand, both the estimation error and the end-to-end latency increase dramatically when the ratio is greater than one. In this case, more than one clusters are simultaneously activated, and hence multiple results are generated [8]. The proposed approach exhibits slower degradation in both the estimation error and the end-to-end latency than the static clustering approach.

[8] Average of multiple estimation results for the same event is used for evaluating the estimation error at the sink.

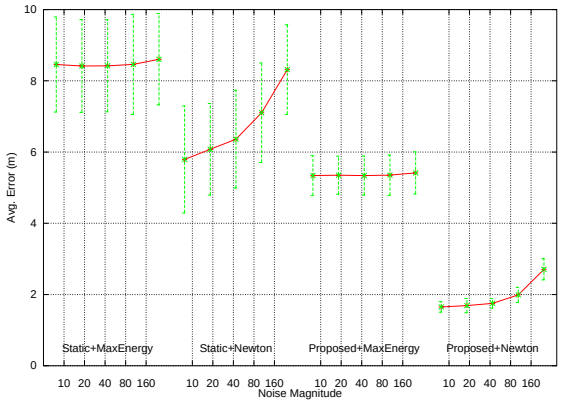


Figure 3.8: Performance comparison of the proposed clustering and static clustering approaches combined with different localization methods with respect to different noise magnitudes. "MaxEnergy" denotes the Voronoi diagram based localization method and "Newton" denotes the nonlinear optimization based method.

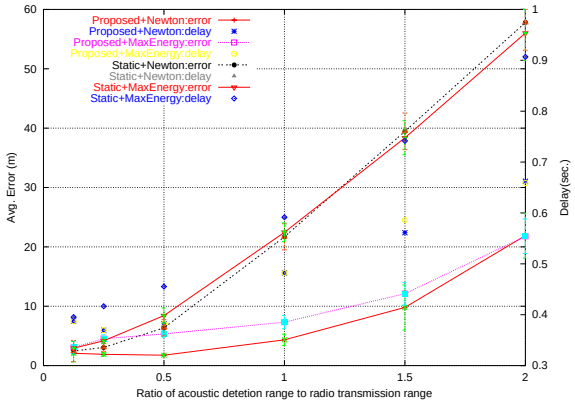


Figure 3.9: Effect of different ratios of the acoustic detection range to the radio transmission range on the performance of the proposed approach (with the 95 % confidence interval).

The effect of varying magnitude of the target We assume the magnitude of the signal from the target is constant in the previous sets of simulations. In this set of simulations, we model the magnitude of the signal, a_v, from the target moving with the speed of v as:

$$a_v = (a_s + k \cdot v) \cdot (1 + \mathcal{N}(0, Var)), \tag{3.7}$$

where a_s is the magnitude of the signal when the target stays static. We assume that the magnitude of the signal increases in proportion to the speed of the target, v. We also figure in a Gaussian random variable with zero mean and Var variance. Given $n+1$ pairs of readings (n of which are from its neighboring sensors, and one from itself), an active CH adjusts its detected magnitude of the signal from the target using Eq. (3.4). Fig. 3.10 gives the result of the proposed clustering approach

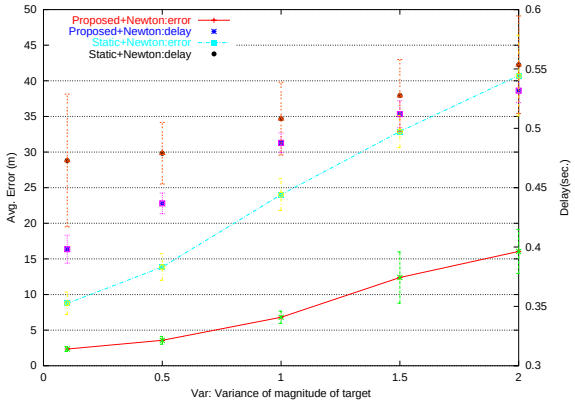


Figure 3.10: The performance of varying magnitude of the target under different variance values (with the 95 % confidence interval).

and the static clustering approach (combined with the nonlinear optimization localization method) with respect to different values of Var. (The noise magnitude is still fixed at 40 and the maximum speed equal to 20 m/s.) Although the performances of the proposed approach degrades with the increase in the noise variance, it behaves more robustly for a wide range of noise magnitudes.

3.5 Summary

In the chapter, we devise and evaluate a fully decentralized, light-weight, dynamic clustering algorithm for target tracking. We envision a hierarchical sensor network that is composed of (a) a static backbone of sparsely placed CHs that assume the role of leaders upon triggered by certain signal events; and (b) moderately to densely populated low-end sensors whose function is to provide sensor information to CHs upon request. A cluster is formed and a CH becomes active, when the acoustic signal strength detected by the CH exceeds a pre-determined threshold. The active CH then broadcasts an information solicitation packet, asking sensors in its vicinity to join the cluster and provide their sensor information. We address and devise solution approaches to the issues (**I1**)–(**I3**) outlined in Chapter 1. Through both probabilistic analysis and *ns-2* simulation, we show with the use of Voronoi diagram, the CH that is closest to the target is (implicitly) selected as the leader and that the proposed dynamic clustering algorithm effectively eliminates contention among sensors and renders more accurate estimates of target locations. As compared to the performance of the static cluster architecture, the proposed approaches reduce the estimation error by 37%. By combining a more sophisticated localization method, the performance in the estimation error can be further improved to be 71%.

We have implemented a subset of the proposed protocols on a lab testbed that consists of Berkeley motes integrated with PC 104 boards. The results will be presented in Chapter 6. One challenge not addressed yet in this chapter is the information dissemination to sinks. In the following chapters, utility-based approaches are used to control the source rate of sensors based on the value of packets generated from the sensors.

Chapter 4

An Energy-Aware Data-Centric Generic Utility Based Approach

Distinct from wireless ad hoc networks, wireless sensor networks are data-centric, application-oriented, collaborative, and energy-constrained in nature. As a result of the unique characteristics of sensor networks, conventional routing and flow control protocols that focus on maximizing raw data throughput and achieving fairness are no longer well suited for sensor networks. Instead, data-centric, utility based approaches that differentiate the treatments of packets with respect to their different values and at the same time, take into account of energy consumption are more adequate.

In this chapter, we formulate the problem of data transport in sensor networks as an optimization problem whose objective function is to maximize the amount of information (utility) collected at sinks (subscribers), subject to the flow, energy and channel bandwidth constraints. In particular, we introduce energy constraints and the notion of quality of data into the formulation. Also, based on a Markov model extended from [6], we derive the link delay and the node capacity in both the single and multi-hop environments, and figure them in the problem formulation. As a result, the resulting solution approach will solve simultaneously the problems of maximizing utility and mitigating congestion.

We show that the formulated optimization problem is general enough to encompass a wide variety of applications in sensor networks, each with a different objective function and subject to different constraints. Specifically, we consider six design dimensions and adapt the generic formulation to meet the various needs of different applications. Following that we either modify the objective function or relax one of the constraint functions to conduct three special case studies under the generic problem formulation. In particular, we show in the first two case studies that

the issue of routing in an environment monitoring system (which was considered in [11]) and the bandwidth allocation problem (which was considered in [62, 86]) can be treated as special cases under the generic problem formulation. In the third case study, we derive an energy aware flow control solution, and investigate via *ns-2* simulation its performance. The simulation results show that as compared with the Ad hoc On Demand Distance Vector (AODV) routing and load balancing routing, the solution derived under the proposed approach achieves higher utility and incurs lower latency.

The rest of the chapter is organized as follows. We present the generic problem formulation in Section 4.1 and the three case studies in Section 4.2. Then, we derive in Section 4.3 the link delay and the node capacity that are necessary in the problem formulation. Following that, we present the simulation results for the third case study — the energy aware flow control problem — in Section 4.4.

4.1 Problem Formulation

In this section we formulate a general utility-based optimization problem that can be tailored to fulfill various goals and requirements for different applications in sensor networks. Before delving into the problem formulation, we state the assumptions made in the thesis:

(A1) Spatial redundancy is not considered: we assume that the sensing data collected from sensors at different locations contributes additive utilities. In reality, surplus sensors may be deployed in the sensing area and the information collected by neighboring sensors may be redundant and correlated. Clustering techniques such as GAF [85] or SPAN [12] have been proposed to group sensors into groups and coordinate activities among them, such that only one sensor needs to be awake in each group to maintain network connectivity and to carry out the sensing task. The data collected in different groups is likely non-redundant.

(A2) The utility of data packets originated from the same node is represented by a single utility function, in spite of the fact that they may be routed along different paths to the sinks.

(A3) The communication cost between sinks is negligible. Once data packets arrives at any of the sinks, they may be relayed to other sinks perhaps via a wireline network, and hence the communication cost is minimal.

The optimization problem is formulated to maximize the total utility of data collected at sinks throughout the system lifetime, subject to the flow constraint, the energy constraint and the channel capacity constraint. For notational convenience, we define the following notions:

- $\mathcal{U}_s(\cdot)$: the utility function that specifies the commodity generated from a sensor s and sent to a sink (perhaps through multiple routes);
- S_n and S_i: the set of sensors and sinks in the sensing field;
- N_k: the set of one-hop neighbors of node k;
- $q_{ij}^{(s)}$: the rate of the commodity s that passes from node i to node j;
- x_i: the source rate originated from node i;
- $x_i^{(s)}$: the source rate of commodity s originated from node i; As the commodity s only originates from node s, $x_i^{(s)} = x_i$ if $i = s$; otherwise, $x_i^{(s)} = 0$;
- E_i: the amount of energy initially equipped with node i;
- e_i: the energy consumed in the idle state per unit time;
- e_s and e_r: the additional energy consumed in transmitting and receiving one unit of data rate per unit time;
- $\overline{d_s}$: the average end-to-end latency that a packet experiences from a sensor s to a sink;
- T: the system lifetime defined as the time interval till the first failure of a node due to the depleted power;
- C_i: the channel capacity of node i; we will derive this value in Section 4.3.

Given the above notations, the problem can be formulated as a nonlinear programming problem as follows:

$$\max_{\mathbf{x},\mathbf{q},T} \quad [\sum_{s\in S_n} \mathcal{U}_s(\sum_{i\in S_i} \sum_{k:i\in N_k} q_{ki}^{(s)}, \overline{d_s})] \cdot T \tag{4.1}$$

$$s.t. \quad \sum_{k:i\in N_k} q_{ki}^{(s)} + x_i \geq \sum_{j:j\in N_i} q_{ij}^{(s)}, \quad \forall i \in S_n, \quad s \in S_n \tag{4.2}$$

$$\{e_r \cdot \sum_{k:i\in N_k} \sum_{s\in S_n} q_{ki}^{(s)} + e_s \cdot \sum_{j:j\in N_i} \sum_{s\in S_n} q_{ij}^{(s)} + e_i\} \cdot T \leq E_i, \quad \forall i \in S_n \tag{4.3}$$

$$\sum_{j:j\in N_i} \sum_{s\in S_n} q_{ij}^{(s)} \leq C_i, \quad \forall i \in S_n \tag{4.4}$$

The objective is to maximize the utility of all received packets within the system lifetime over a vector of source rates of commodities ($\mathbf{x}$), a vector of link flow ($\mathbf{q}$) and the system lifetime (T). As such, the objective function (Eq. (4.1)) is expressed as the product of the system lifetime and the utility of all commodities received at the sinks per unit time. The utility function for the commodity s is a function of the total rate of the commodity s arriving at sinks and the average end-to-end latency it sustains. By assumptions (**A2**) and (**A3**), the rate of commodity s arriving at sinks is the sum of all the incoming flows of commodity s to any of sinks. Since flows travelling through different routes to sinks endure different latencies, we express the utility function as a function of the average latency $\overline{d_s}$ to account for the average loss of utility due to the delay. Moreover, with different qualities of data, the different quantized utility functions (such as in [48]) can be used to evaluate the utility of a data packet.

The first constraint (Eq. (4.2)) is the flow conservation. The sum of both the incoming flows of commodity s and the flow of commodity s originated from a node is greater than or equal to the sum of the outgoing flows of commodity s, with the inequality implying that intermediate relay nodes may drop packets they forward. The second constraint (Eq. (4.3)) is the energy constraint, while the third constraint (Eq. (4.4)) is the capacity constraint, i.e., the sum of the outgoing flows of all the commodities from a node i should be less than its channel capacity C_i (the value of which will be derived in Section 4.3).

The problem formulated above aims to maximize the total utility received at the sinks, by controlling the parameter vectors $\mathbf{x}$ and $\mathbf{q}$ (which in turn are related to flow control and routing decisions). As a matter of fact, the above problem formulation encompasses a wide variety of requirements and objectives for different applications in sensor networks. In what follows, we discuss six possible design dimensions and their corresponding amendments to the above problem formulation:

1. *Flow conservation*: If intermediate relay nodes are not allowed to discard packets they forward, the inequality in Eq. (4.2) is changed to an equality. With the flow conservation constraint, for each commodity s, the sum of the incoming flows of commodity s at sinks is equal to the rate x_s originated at node s, and hence the objective function in Eq. (4.1) can be rewritten as:

$$\max_{\mathbf{x},\mathbf{q},T} \quad [\sum_{s \in S_n} \mathcal{U}_s(x_s, \overline{d_s})] \cdot T \tag{4.5}$$

2. *Flow indivisibility constraint*: If a commodity from a sensor node s cannot be routed through multiple paths, an additional constraint has to be added in Eq. (4.2) such that for each commodity s, only one incoming and one outgoing flow has positive rate and others are zero. This makes it more difficult to solve the optimization problem because of its discrete constraint.

3. *Flow control*: In the problem formulation, both the routing and flow control problems are jointly considered. An alternative approach is to solve the optimization problem in two steps. The routes are determined first by a routing protocol and figured into the optimization problem. The optimization problem then solves the flow control problem, by optimizing the total utility over the vector of source rates, $\mathbf{x}$.

4. *Quality-driven utility function*: If the quality of data is considered, the utility function of each sensor is determined based on the quality of the data sensed; otherwise, the utility functions are the same for all the sensors.

5. *Effect of latency on utility*: Whether the latency affects the utility of the data sensed is application-dependent. In general, the utility of data decays with the latency but the decay function (convex, linear, or concave) varies with the application characteristics. Alternatively, the effect of latency can be figured in as a constraint into the optimization problem.

6. *Energy awareness*: If the energy constraint is not considered, the problem formulation can be simplified as follows: The system lifetime T can be removed from the objective function and the energy constraint (Eq. (4.3)) can be removed.

Whether or not a utility-based approach is effective is contingent upon the proper design of the utility function. Several rules can be applied to determine appropriate utility functions in sensor networks: (i) More data generates more utility, but the marginal utility decreases due to spatial and temporal redundancy; (ii) The utility of data decreases with the latency; (iii) Better data quality results in higher utility. Based on the above rules, we characterize the utility functions as follows:

$$\mathcal{U}_s(x_s, \overline{d_s}) = a_s \cdot \mathcal{U}(x_s), \cdot \mathcal{D}_s(\overline{d_s}) \tag{4.6}$$

where a_s is a parameter that determines the importance of, and the quality of, data originated from sensor s. The function $\mathcal{U}(\cdot)$ is a non-decreasing, concave utility function of the source rate.

The function $\mathcal{D}_s : \mathcal{R}^+ \rightarrow [0, 1]$ is a non-increasing utility decay function of the average end-to-end latency, $\overline{d_s}$, of packets from sensor s. $\mathcal{D}_s(\cdot)$ is application-dependent, and can be convex, linear, concave functions or a combination thereof. For mission-critical applications, the value of utility decays abruptly after the system tolerance period and is hence a convex function in that region. On the other hand, for non-mission-critical applications, the utility is not significantly affected by the latency and can thus be expresses as a concave function. The average end-to-end latency of packets from sensor i can be estimated as

$$\overline{d_s} = \sum_{j,k} q_{jk}^{(s)} \cdot h_j / \sum_{j,k} q_{jk}^{(s)}, \tag{4.7}$$

where h_j is the relay latency from node j to its next hop, including both the link and queueing delays. Both values will be derived, based on a Markov model and a *M/D/1* queueing model, respectively, in Section 4.3.

4.2 Application Examples Under the Problem Formulation

In this section, we consider three representative problems that have been considered in the literature, and show that they are special cases of the general problem formulation given in Section 4.1.

4.2.1 Case Study I: Routing in a Environment Monitoring System

Consider an environment monitoring system in which sensor nodes monitor environmental changes such as temperature, moisture, habitat, chemical contaminant or construction safety. Sensors that are deployed in an area periodically send their sensor readings back to the control center, so that data can be logged and/or further analyzed. The sending rate $\mathbf{x}$ is usually given, and the utility of data from all sensors is the same and does not decay with the latency. As the objective of such a system is to maximize the system lifetime under the condition that the sensors cover the entire area, the objective function (Eq. (4.1)) is modified as $\{\max_{\mathbf{q}} T\}$, with the constraints in Eqs. (4.2)–(4.4) remaining the same.

This formulation is similar to that in [11], with a major difference: the formulation in [11] only considers the transmission power consumption, but not the node capacity constraint. Without considering the node capacity constraint, the problem was reduced to a linear programming problem (that may render an infeasible solution).

4.2.2 Case Study II: Flow Control

The problem formulated in Section 4.2.1 assumes that the source rate of commodities $\mathbf{x}$ has been given and considers the routing problem by determining $\mathbf{q}$ to maximize the system lifetime. On the other extreme, one can assume that the routes are given for each pair of source and destination and maximizes the utility of received packets at sinks by controlling the vector of source rate $\mathbf{x}$. This is termed as the flow control problem (a.k.a. the bandwidth allocation problem). Several efforts have been made along this research avenue: Kelly *et al.* [40] proposed a price-based bandwidth allocation approach to maximize the total utility of all users in wireline environments. Xue *et al.* [86] applied the same formulation to ad hoc networks. In both work only the constraint on the link capacity is considered. Note that although considering the constraint on the link capacity is adequate in wireline environments, it may not be sufficient in ad hoc environments. Instead the constraint on the node capacity should be considered, as all outgoing links from a node share the same channel. The constraint on the link capacity suffices only when a node is equipped with multiple transceivers operating at different channels or with directional antennas. Qiu *et al.* [62] considered the constraint on the node capacity and further relaxed the flow conservation rule.

All the above work does not consider the energy constraint and thus can only be applied to sensors that come with tethered power supplies. Without the energy constraint, the system lifetime, T, is no longer a control variable and the objective function becomes separable, rendering a separable nonlinear programming problem. Note that the dual problem of a separable nonlinear programming problem can be readily evaluated [4], and a distributed solution can also be derived [52].

4.2.3 Case Study III: Energy-Aware Flow Control

In sensor networks, energy saving is the top priority for system design. Hence, the problem formulation in Section 4.2.2 can be augmented by adding the energy constraint and figuring in the system lifetime in the objective function (as in Eq. (4.5)). Recall that the optimization problem in Eq. (4.5) considers both routing and flow control decisions together, and is a NP-hard problem as shown in [81]. A simple solution of the augmented problem can be derived into two phases. In the first phase, based on the geographical information, a load balancing route is first determined for each sensor using the algorithm given in Fig. 4.1. The algorithm rests on the assumption that a link $l_{i\to j}$ is in G if the distance between i and j is less than the radio range and j is closer

1. $G \leftarrow S_i$
2. $Q \leftarrow S_n$ /*sorted in the increasing order of distance to any one of sinks*/
3. while ($Q \neq \emptyset$)
4. $\quad s \leftarrow Q.head()$; $s.cost \leftarrow 0$
5. $\quad$ Run Dijkstra's algorithm to find the shortest path p from s to its closest sink using nodes $\in G$
6. $\quad$ Backtrack the path p from the sink to s and $\forall$ nodes $v \in p \backslash S_i$, $v.cost \leftarrow v.cost + 1$
7. $\quad G \leftarrow G \bigcap \{s\}$

Figure 4.1: The routing algorithm that balances loads based on geographical information.

to the destination than i (greedy geographical routing). It finds routes for sensors, starting from the sensor that is closest to any of the sinks. The cost associated with the node represents the number of flows it originates or forwards for other nodes. Since a route is assigned for each sensor sequentially, when a node further away from a sink determines its route, it attempts not to select nodes with high costs. Therefore, the objective of load balance can be achieved, and this routing algorithm serves as a good basis for finding a good solution for the optimization problem. The resulting routes derived in the algorithm are expressed in the routing matrix R, with $R_{sf} = 1$ if the node f is on the route of sensor s to its closest sink; and $R_{sf} = 0$ otherwise.

In the second phase, given the set of routes for all the sensors and under the assumption that the utility of data does not decay with the latency, the optimization problem becomes (let I and $\mathbf{e}$ be identity matrix and entity vector, respectively):

$$\begin{aligned} \max_{\mathbf{x},T} \quad & [\sum_{s \in S_n} \mathcal{U}_s(x_s)] \cdot T \\ s.t. \quad & [e_s \cdot R^T \cdot \mathbf{x} + e_r \cdot (R^T - I) \cdot \mathbf{x} + e_i \cdot \mathbf{e}] \cdot T \leq \mathbf{E}, \\ & R^T \cdot \mathbf{x} \leq \mathbf{C}. \end{aligned} \tag{4.8}$$

Recall that $\mathbf{x}$, $\mathbf{E}$, and $\mathbf{C}$ are the vectors of source rate, initial energy, and node capacity for sensors, respectively. The vector $\mathbf{x}$ is the control variable, $\mathbf{E}$ is given by the system, and $\mathbf{C}$ is a topology-dependent parameter and will be derived in Section 4.3.

Unfortunately, both the objective function and the first constraint are not concave functions. As a result, a solution satisfying Karush-Kuhn-Tucker conditions may not be the global optimal solution. However, the above solution gives a lower bound for the optimal solution, and can be used as an approximate solution. Several global optimization methods without searching the global optimal solution exhaustively have been proposed. Vanderbei *et al.* [78] propose an efficient approach for non-convex nonlinear programming by slightly modifying the interior-point approach for quadratic programming. The interested reader is referred to [34] for several global optimization

methods. In the thesis, we use $MATLAB^{®}$ to find the solution by attempting on several random initial points. The performance of the solution under this problem formulation will be evaluate in Section 4.4.

4.3 Derivation of Link Delay and Node Capacity in 802.11-like MAC Protocol

In the general optimization problem (Eqs. (4.1)-(4.4)), the average end-to-end latency is needed in Eq. (4.1) and the node capacity is needed in Eq. (4.4). In this section, we derive, based on a Markov chain model, the link delay in the case of single hop networks. Next, we obtain the link delay under the multi-hop environments by considering the flow contention graph. Furthermore, we *approximate* the queueing model of relay nodes as *M/D/1* queue to estimate the node capacity and end-to-end latency. We assume that an IEEE 802.11-like backoff mechanism has been used as the underlying MAC protocol.

In the 802.11-like backoff mechanism, before a node first attempts to send a packet, it sets a backoff timer uniformly distributed in $[0, W_0]$, where $W_0 = W$ is the initial maximum contention window. A node counts down its backoff timer whenever the channel is sensed idle; otherwise, it freezes the timer. When the value of the backoff timer becomes zero, the node transmits the packet. If the packet collides with other packets, i.e. the sender does not receive an acknowledgment, the backoff timer is reset to be uniformly distributed in $[0, W_i]$, where the index i represents the number of retransmissions, and $W_i = 2^i \cdot W_0 = 2^i \cdot W$ if the exponential binary backoff policy is adopted. The *maximum backoff state* is m, i.e., the maximum window size $W_{max} = 2^m \cdot W$.

4.3.1 Derivation of Link Delay in the Single Hop Case

The derivation of the link delay within one hop is based on the model in [6]. Two major assumptions (approximations) are made in our model (and also in [6]). First, the network consists of M backlogged nodes within the one-hop transmission range of each other. That is, each node in the system can hear each other and always has packets to send. Second, the time after detecting an idle channel is slotted, and in each transmission attempt the probability, p, that a packet collides with some other(s) in a slot is a constant, regardless of the number of retransmissions the packet has incurred. The probability p is referred to as *conditional collision probability*. The model can be described with a Markov chain shown in Fig. 4.2, with the state being a 2-tuples $(s(t), b(t))$, where

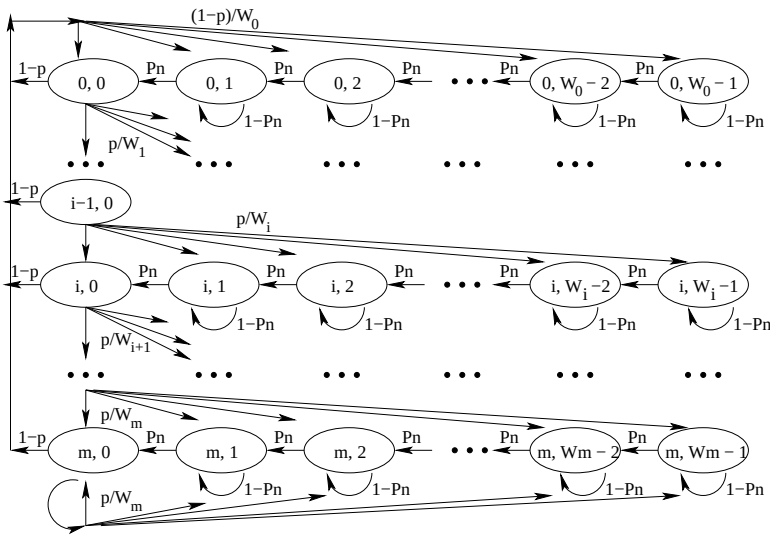


Figure 4.2: The Markov chain model for analyzing the link delay under an IEEE 802.11-like backoff mechanism.

$s(t)$ is the retransmission times (ranging from 0 to m, the maximum backoff state) and $b(t)$ is the value of the backoff timer of a node at time t.

The major difference between our model and that in [6] lies in that the later does not consider the effect of freezing the backoff timer when a node in its backoff period senses the channel busy. To account for this effect, there are two state transitions from state (i, j) $(j > 0)$ to other states in our model: (i) one is with the probability of sensing the channel idle, p_n, i.e., the other $M - 1$ nodes do not send packets, and the state becomes $(i, j - 1)$ as a result of the backoff timer being decremented by one; and (ii) the other one is with the probability of sensing the channel busy, $1 - p_n$, i.e., at least one other node sends a packet in the slot, and the state remains at (i, j). Note that p_n is the conditional probability of sensing the channel idle given that the node itself does not send any packet (note that $j > 0$) in the current time slot, and hence $p_n = (1 - \tau)^{M-1}$, where τ is the probability that a node transmits a packet in a slot time.

By the operations of the backoff mechanism, the state transition probabilities can be expressed as follows:

$$\begin{cases} P\{i,k|i,k\} = 1 - p_n, & k \in (1, W_i - 1) \quad i \in (0, m), \\ P\{i,k-1|i,k\} = p_n, & k \in (1, W_i - 1) \quad i \in (0, m), \\ P\{0,k|i,0\} = (1-p)/W_0, & k \in (0, W_0 - 1) \quad i \in (0, m), \\ P\{i,k|i-1,0\} = p/W_i, & k \in (0, W_i - 1) \quad i \in (1, m), \\ P\{m,k|m,0\} = p/W_m, & k \in (0, W_m - 1). \end{cases} \tag{4.9}$$

Let $b_{i,k} = \lim_{t\to\infty} P\{s(t) = i, b(t) = k\}, i \in (0, m), k \in (0, W_i - 1)$ be the stationary distribution of state (i, k). Given the Markov model in Fig. 4.2 and the state transition probabilities in Eq. (4.9), the equilibrium equations for the states $(i, W_i - 1), (i, k), 0 < k < W_i - 1$, and $(i, 0)$ in the case that the number of retransmissions is $i, 0 < i < m$ are:

$$\begin{cases} b_{i,W_i-1} = \frac{p}{W_i} \cdot b_{i-1,0} + (1-p_n) \cdot b_{i,W_i-1} & \Rightarrow \quad b_{i,W_i-1} = \frac{p}{W_i \cdot p_n} \cdot b_{i-1,0}, \\ b_{i,k} = p_n \cdot b_{i,k+1} + \frac{p}{W_i} \cdot b_{i-1,0} + (1-p_n) \cdot b_{i,k} & \Rightarrow \quad b_{i,k} = b_{i,k+1} + \frac{p}{W_i \cdot p_n} \cdot b_{i-1,0} = \frac{(W_i-k)p}{W_i \cdot p_n} \cdot b_{i-1,0}, \\ b_{i,0} = \frac{p}{W_i} \cdot b_{i-1,0} + p_n b_{i,1} = p b_{i-1,0} = p^i \cdot b_{0,0} \end{cases} \tag{4.10}$$

Similarly, the equilibrium equations for the states $(m, W_m - 1), (m, k), 0 < k < W_m - 1$, and $(m, 0)$ in the case that the number of retransmissions is equal to or more than the maximum backoff stage m become:

$$\begin{cases} b_{m,W_m-1} = \frac{p}{W_m}(b_{m-1,0} + b_{m,0}) + (1-p_n) b_{m,W_m-1} & \Rightarrow \quad b_{m,W_m-1} = \frac{p}{W_m \cdot p_n}(b_{m-1,0} + b_{m,0}), \\ b_{m,k} = \frac{p}{W_m}(b_{m-1,0} + b_{m,0}) + p_n \cdot b_{m,k+1} + (1-p_n) b_{m,k} & \Rightarrow \quad b_{m,k} = \frac{(W_m-k)p}{W_m \cdot p_n}(b_{m-1,0} + b_{m,0}), \\ b_{m,0} = \frac{p}{W_m}(b_{m-1,0} + b_{m,0}) + p_n \cdot b_{m,1} & \Rightarrow \quad b_{m,0} = \frac{p}{1-p} \cdot b_{m-1,0} = \frac{p^m}{1-p} \cdot b_{0,0}. \end{cases} \tag{4.11}$$

Finally, the equilibrium equations for the states $(0, W_0 - 1), (0, k), 0 < k < W_0 - 1$, and $(0, 0)$ in the case that the packet being sent incurs no retransmission are:

$$\begin{cases} b_{0,W_0-1} = (1-p_n) b_{0,W_0-1} + \frac{1-p}{W_0} \cdot \sum_{i=0}^{m} b_{i,0} & \Rightarrow \quad b_{0,W_0-1} = \frac{1-p}{W_0 \cdot p_n} \cdot \sum_{i=0}^{m} b_{i,0}, \\ b_{0,k} = (1-p_n) \cdot b_{0,k} + p_n \cdot b_{0,k+1} + \frac{1-p}{W_0} \cdot \sum_{i=0}^{m} b_{i,0} & \Rightarrow \quad b_{0,k} = \frac{(1-p)\cdot(W_0-k)}{W_0 \cdot p_n} \cdot \sum_{i=0}^{m} b_{i,0}, \\ b_{0,0} = p_n \cdot b_{0,1} + \frac{1-p}{W_0} \cdot \sum_{i=0}^{m} b_{i,0} & \Rightarrow \quad b_{0,0} = (1-p) \cdot \sum_{i=0}^{m} b_{i,0}. \end{cases} \tag{4.12}$$

The stationary probabilities of state $(i, 0)$ and (i, k) in Eqs. (4.10)-(4.12) can be expressed in terms of $b_{0,0}$ as:

$$b_{i,0} = b_{0,0} \cdot \begin{cases} p^i, & 0 < i < m, \\ \frac{p^m}{1-p}, & i = m, \end{cases} \tag{4.13}$$

and

$$b_{i,k} = \frac{(W_i - k)}{W_i \cdot p_n} \cdot b_{0,0} \cdot \begin{cases} 1, & i = 0, & 0 < k < W_0, \\ p^i, & 0 < i < m, & 0 < k < W_i, \\ \frac{p^m}{1-p}, & i = m, & 0 < k < W_m. \end{cases} \tag{4.14}$$

$b_{0,0}$ can be derived by equating the sum of the probabilities of all states to one, i.e.,

$$1 = \sum_{i=0}^{m} \sum_{k=0}^{W_i - 1} b_{i,k} \Rightarrow b_{0,0} = \frac{2p_n(1-p)(1-2p)}{(2p_n - 1)(1-2p) + W(1-p-2^m p^{m+1})}. \tag{4.15}$$

Since a node only attempts to send a packet when its backoff timer is zero, we can express the probability τ that a node transmits a packet in a randomly chosen slot as the sum of state probabilities with $b(t) = 0$:

$$\tau = \sum_{i=0}^{m} b_{i,0} = \frac{b_{0,0}}{1-p} = \frac{2p_n(1-2p)}{(2p_n - 1)(1-2p) + W(1-p-2^m p^{m+1})}. \tag{4.16}$$

Note that τ is expressed in terms of the conditional collision probability, p. On the other hand, the conditional collision probability p is the probability that a packet being sent collides with packets from other nodes. That is, p can be expressed in terms of transmission probability τ as follows:

$$p = 1 - (1-\tau)^{M-1} = 1 - p_n. \tag{4.17}$$

With Eqs. (4.16) and (4.17) we may numerically solve both τ and p. The average number of attempts to transmit a packet can be expressed in terms of p:

$$\text{Ave. \# of attempts} = \sum_{i=0}^{\infty} i(1-p)p^{i-1} = \frac{1}{1-p}. \tag{4.18}$$

With all the state probabilities derived, we now derive the link delay — the expected latency to transmit a packet to the next hop. Let $d_{i,k}$ be the expected time[1] to transmit a packet when a node is at the state (i,k). By the Markov model given in Fig. 4.2, there are two state transitions from state $(i,k), k \neq 0$. Either the channel is idle or busy in the next slot. In the latter case, the node remains at state (i,k) for a time interval, $\overline{T_f}$, that is equal to the conditional expected freeze time given that the channel is busy.

$$d_{i,k} = p_n \cdot (1 + d_{i,k-1}) + (1-p_n) \cdot (\overline{T_f} + d_{i,k}) \Rightarrow d_{i,k} = k \cdot [1 + \frac{(1-p_n) \cdot \overline{T_f}}{p_n}] + d_{i,0}, \quad \forall 0 \leq i \leq m, 0 < k < W_i \tag{4.19}$$

[1] The unit of $d_{i,k}$ is the slot time.

The conditional expected freeze time $\overline{T_f}$ can be expressed as

$$\overline{T_f} = \frac{(M-1)\tau(1-\tau)^{M-2}}{1-p_n} \cdot T_s + \frac{1-(1-\tau)^{M-1}-(M-1)\tau(1-\tau)^{M-2}}{1-p_n} \cdot T_c, \tag{4.20}$$

where the first term represents the case when only one node out of the other $M-1$ nodes sends a packet, the packet is successfully transmitted, and the channel is occupied for an interval of T_s, i.e., the expected time to successfully transmit a packet; and the second term represents the case when more than one node attempt to send packets, the packets collide with each other and the channel is busy for an interval of T_c, i.e., the expected collision period sensed by a node.

The values of T_s and T_c vary depending on whether or not the RTS/CTS mechanism is used. In the basic mode (denoted as bas), the RTS/CTS mechanism is not used, T_s^{bas} includes the packet header (H), the expected packet payload ($E[P]$), short inter-frame space ($SIFS$), acknowledgment (ACK), distributed inter-frame space ($DIFS$) and twice of the propagation delay (δ); and T_c^{bas} contains the packet header (H), the expected length of the longest packet payload involved in a collision ($E[P^*]$), $DIFS$, and δ. In the thesis we assume all packets have the same payload size, and thus $E[P] = E[P^*] = P$. T_s^{bas} and T_c^{bas} are expressed as:

$$\begin{cases} T_s^{bas} = H + E[P] + SIFS + \delta + ACK + DIFS + \delta, \\ T_c^{bas} = H + E[P^*] + DIFS + \delta. \end{cases} \tag{4.21}$$

In the RTS/CTS mode (denoted as rts), the service time, T_s^{rts}, contains several additional terms, i.e., request-to-send packet, (RTS), clear-to-send packet (CTS), $2 \times SIFS$, and $2 \times \delta$, as compared with the basic mode. On the other hand, T_c^{rts} contains only RTS, $DIFS$, and δ.

$$\begin{cases} T_s^{rts} = RTS + SIFS + \delta + CTS + SIFS + \delta + H + E[P] + SIFS + \delta + ACK + DIFS + \delta, \\ T_c^{rts} = RTS + DIFS + \delta. \end{cases} \tag{4.22}$$

By the Markov model, state transitions of state $(i, 0)$ are different from those of state $(i, k), k \neq 0$. At state $(i, 0)$, a node attempts to send a packet, and incurs either a successful transmission (that takes time T_s), or a collision (that causes the channel to be busy for an interval of T_c and the backoff stage to increase by one). The expected time to transmit a packet for state $(m, 0)$ and

state $(i,0), 0 \leq i < m$ are respectively

$$\begin{cases} d_{m,0} = (1-p)T_s + \frac{p}{W_m}\sum_{k=0}^{W_m-1}(d_{m,k}+T_c) & \Rightarrow \quad d_{m,0} = T_s + \frac{p}{1-p}T_c + \frac{p(2^m W-1)}{2(1-p)}[1+\frac{(1-p_n)\overline{T_f}}{p_n}], \\ d_{i,0} = (1-p)T_s + \frac{p}{W_{i+1}}\sum_{k=0}^{W_{i+1}-1}(d_{i+1,k}+T_c) & \forall\, 0 \leq i \leq m-1. \end{cases} \tag{4.23}$$

Combining Eq. (4.19) and (4.23) and using induction from $m-1$ to 0, we can express $d_{i,0}, 0 \leq i < m$ as follows:

$$d_{i,0} = T_s + \frac{p}{1-p}\cdot T_c + \frac{p}{2(1-p)}\cdot[1+\frac{(1-p_n)\overline{T_f}}{p_n}]\cdot[\frac{2^{i+1}pW(1-(2p)^{m-i-1})}{1-2p} + 2^{i+1}W - 1]\ , \forall 0 \leq i < m. \tag{4.24}$$

Since the initial state is among the state $(0,0)$ to $(0,W-1)$, the average link delay D_l is the average time to transmit a packet with the average taken over state $(0,0)$ to state $(0,W-1)$. Thus,

$$D_l = \frac{1}{W}\cdot\sum_{k=0}^{W-1} d_{0,k} = T_s + \frac{p}{1-p}\cdot T_c + \frac{1+\frac{(1-p_n)\overline{T_f}}{p_n}}{2(1-p)}\cdot(\frac{pW(1-(2p)^m)}{1-2p} + W - 1). \tag{4.25}$$

After several arithmetic operations, D_l can be expressed in terms of M and τ as follows:

$$D_l = M(T_s - T_c) + \frac{1-\tau}{\tau}\cdot\{1+[(1-\tau)^{-M}-1]\cdot T_c\}. \tag{4.26}$$

Although τ is also a function of M, it is not easy to express τ in terms of M. Instead both τ and p are numerically solved through Eq. (4.16) and Eq. (4.17). The numerical values of D_l under both the basic and RTS/CTS modes are given in Fig. 4.3. under the assumption that the data transmission rate of wireless channel is 40 kbps. Surprisingly the curve of D_l is very close to a linear function of M. This is perhaps attributed in part by the fact that the first term $M(T_s - T_c)$ in Eq. (4.26) dominates the second term. The result is intuitively correct since 802.11 is fair for all backlogged nodes, and each such node takes turns to transmit a packet. As a result the one-hop link delay increases linearly with M. The numerical results of D_l in the single hop case will serve a base for deriving the link delay under the multiple-hop environment.

4.3.2 Derivation of Link Delay in the Multiple Hop Case

The major hurdle in deriving the link delay in the multiple hop case is that the assumption that all nodes can hear each one another no longer holds. The transmission of a node may interfere with that of another node outside its radio range. This is know as the *hidden terminal problem*. In

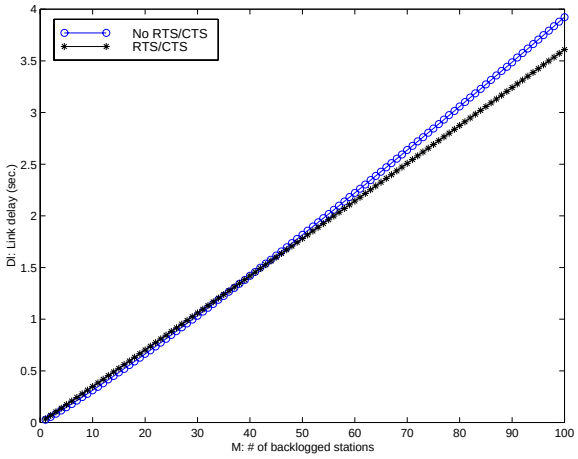


Figure 4.3: The link delay in the basic mode and in the RTS/CTS mode in the one-hop environment.

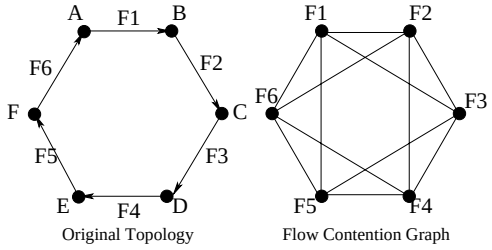


Figure 4.4: An example that shows how the flow contention graph is derived from the original topology. This figure is reprinted from [53].

general, the number of backlogged nodes that compete with a node depends on the locations and the traffic loads of other nodes. The flow contention graph proposed in [53, 86, 37] is commonly used to approximate the number of backlogged nodes with which a node may contend. In the flow contention graph, a vertex and an edge represent, respectively, a flow and the potential contention between two flows. For example, as shown in Fig. 4.4 flow $F1$ may interfere with flows $F2$, $F3$, $F5$, and $F6$ and thus the vertex $F1$ is adjacent to those vertices in flow contention graph.

To infer the exact number of backlogged nodes that compete with a sender node s, node s needs to know the buffer status, the link traffic and the scheduling algorithm at other nodes, which is essentially infeasible. Instead, we observe that two flows contend with each other if either the sender or the receiver of one flow is within the transmission range of either the sender or receiver of the other flow, and approximate the number of potentially competing backlogged nodes as the number of neighbors that lie within the two-hop neighborhood of node s. That is, we estimate the link delay incurred by a flow emanating from node s as $D_l(|N_s| + 1)$, where $|N_s|$ is the number of neighbors within node s's two hops neighborhood. Note that this approximation over-estimates

the number of potentially conflicting backlogged nodes and we err on the pessimistic side.

4.3.3 Derivation of Node Channel Capacity

After deriving the link delay, we now derive the node capacity. Since all the flows emanating from node s share a single interface, we estimate node s's channel capacity C_s as the reciprocity of the link delay $1/D_{l_s}$. Recall that the derivation of link delay in Section 4.3.2 is based on two conditions: every node in the system is backlogged and all two-hops neighbors of a node potentially conflict with itself. Both tend to over-estimated the link delay. To compensate the effect of over-estimation, we need to consider non-backlogged cases, and a queueing analysis is needed to estimate the probability that a node is in the non-backlogged status. We approximate the queueing model at each node as a *M/D/1* queue, in which the probability of backlog and the mean waiting time can be readily derived. The revised node capacity is derived as follows. First, the optimal source rate of each source is solved according to the over-estimated node capacity under the backlogged status. Then the system load ρ_s at node s is calculated as the ratio of all outgoing traffic to its node capacity. Since the probability that the queue at node s is non-empty is equal to ρ_s, the expected link delay is the average link delay over all possible conditions in the number of *busy* neighbors.

$$D_{l_s} = \sum_{i=0}^{|N_s|} D_l(i+1) \cdot [\prod_{j \in B_s, |B_s|=i} \rho_j] \cdot [\prod_{j \in N_s \setminus B_s} (1-\rho_j)], \tag{4.27}$$

where B_s represents the set of busy neighbors of node s within the range of two hops. Each term within the summation represents the conditional link delay given the existence of i busy neighbors, and equals the product of $D_l(i+1)$ and the probability of i busy neighbors. The refined estimation of node capacity C in Eq. (4.4) is the reciprocity of the refined link delay in Eq. (4.27). Also, the end-to-end delay needed in Eq. (4.1) can be calculated as the sum of the relay latency given in Eq. (4.7). With the approximation of *M/D/1* queue, the relay latency at node j can be estimated as:

$$h_j = \frac{1 - \frac{\rho_j}{2}}{C_j(1-\rho_j)} \tag{4.28}$$

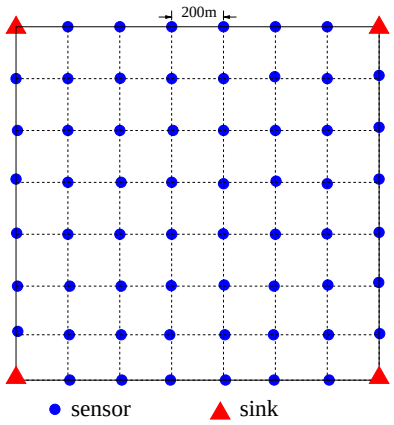


Figure 4.5: The topology used in the simulation study. A total of 4 sinks and 60 sensors are deployed on a square grid. The distance between neighboring sensors is 200 meters, and each sensor has at most 4 neighbors.

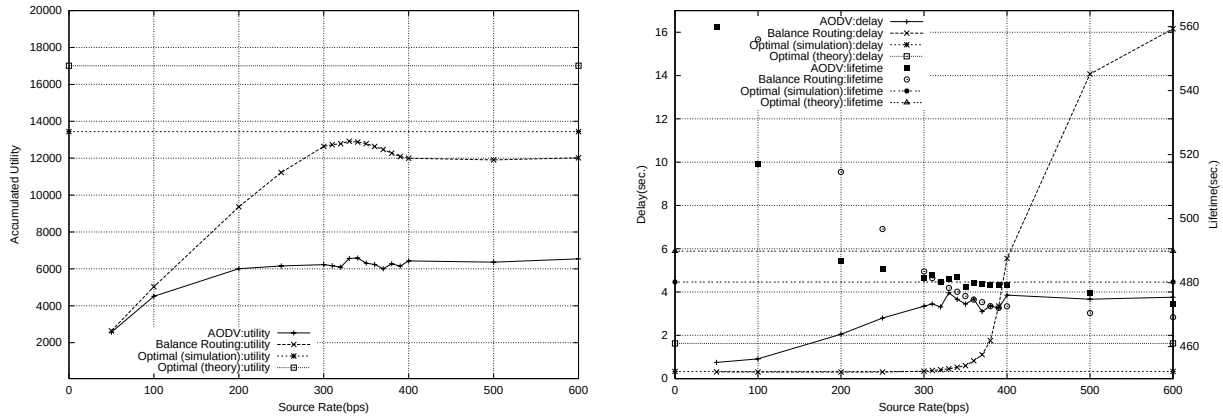
4.4 Performance Evaluation of Energy Aware Flow Control Problem

In this section, we evaluate via *ns-2* simulation the performance of the energy-aware flow control problem (case III) formulated in Section 4.2. We use the topology depicted in Fig. 4.5 in the simulation: 4 sinks and 60 sensors are deployed on a square grid. The distance between neighboring sensors is 200 meters. We assume the radio transmission range is 250 meters. Therefore, each sensor has at most 4 neighbors. The data transmission rate of wireless channel is assumed to be 40 kbps. The optimization problem to be solved is given in Eq. (4.8) in which the routing matrix R is determined by the algorithm given in Fig. 4.1, and the utility function is defined as $\mathcal{U}_s(x_s) = v_s \cdot log(x_s + 1)$, where v_s and x_s are the utility value of a packet and the source rate (in units of #packets/second) for sensor node s, respectively. Function $\mathcal{U}_s$ is a non-decreasing and concave function of node s's sending rate. The energy, E_i, each sensor is initially equipped with is 500 joules. The parameters for energy consumption follows the setting in [12], i.e., the energy consumption incurred in the transmission, reception, and idle state is 1.4, 1.0, and 0.83 W, respectively. Hence, e_i is 0.83 W. e_s and e_r are the additional energy consumed (in addition to e_i) in sending and receiving a packet, and are equal to the product of T_s, the time to send a packet and 0.57(=1.4-0.83) and 0.17(=1.0-0.83), respectively. (Note that the units of e_s and e_r are joules per packet.) The payload size of a packet is set to be 70 bytes (including 20 bytes of IP header but not MAC and PHY headers). With the above parameter setting, the source rate, x_i of each sensor i is obtain by solving Eq. (4.8) using $MATLAB^{®}$.

Comparison with respect to accumulated utility, end-to-end delay and system lifetime: We carry out two sets of simulations. In the first set of simulations, the utility values of packets from all the sensors are assumed to be equal to 1 while in the second set of simulations, the utility value of packets from node s, v_s, is uniformly distributed in $[1, 100]$. We compare the solution derived under the energy-aware flow control problem with two approaches: AODV [60] routing and load balanced routing in Fig. 4.1, under a wide range of constant source rates. Fig. 4.6 gives the accumulated utility, the end-to-end packet delay and system lifetime under the optimal energy-aware flow control, AODV routing, and load balanced routing in the first set of simulations. Curves labeled 'Optimal(simulation)' give simulation results of the optimal source rate derived in Eq. (4.8), while those labeled 'Optimal(theory)' represent theoretical solutions of Eq. (4.8). The best performance of load balanced routing is better than that of AODV with respect to both the accumulated utility and the end-to-end delay. This is in part due to the fact that load balanced routing is based on static information and does not incur routing overhead. Both AODV and balanced routing achieve the highest utility at the source rate of approximately 330 bps[2]. The utilities decrease as the source rate deviates from the optimal point. The derived optimal solution outperforms both AODV and load balance routing with constant source rate with respect to both the accumulated utility and the end-to-end delay. The performance in the system lifetime of each scheme is almost the same. The derived source rates at nodes that are closer to sinks are higher. The capacities of nodes close to sinks or the field boundary are also larger than those of the nodes in the central region because the nodes in the former group have a smaller number of two-hops neighbors. Another interesting observation is that there exists a gap between simulation and theoretical results. The later gives higher utility, higher system lifetime and longer end-to-end latency. The first two facts result from the under-estimated energy consumption model because the energy consumption model in Eq. (4.8) does not take the energy consumed in overhearing and retransmission into account. On the other hand, the derived end-to-end delay is based on the link delay derived under conservative conditions.

Fig. 4.7 gives the accumulated utility, the end-to-end packet delay and the system lifetime under the optimal energy-aware flow control, AODV routing, and load balanced routing in the second set of simulations. The results exhibit similar trends as those in the first set of simulation, except that the performance gain of the derived solution is significantly increased. The flow control solution achieves higher utility than balanced routing by 30%, due to the fact that different utility values

[2]The source rate is evaluated based on pure data payload but not overhead in IP and MAC headers.



(a) (b)

Figure 4.6: Performance comparison of the energy-aware flow control solution against AODV routing and load balanced routing with respect to (a) utility and (b) end-to-end packet delay and system lifetime under a wide range of source rates. The utility values of packets are the same.

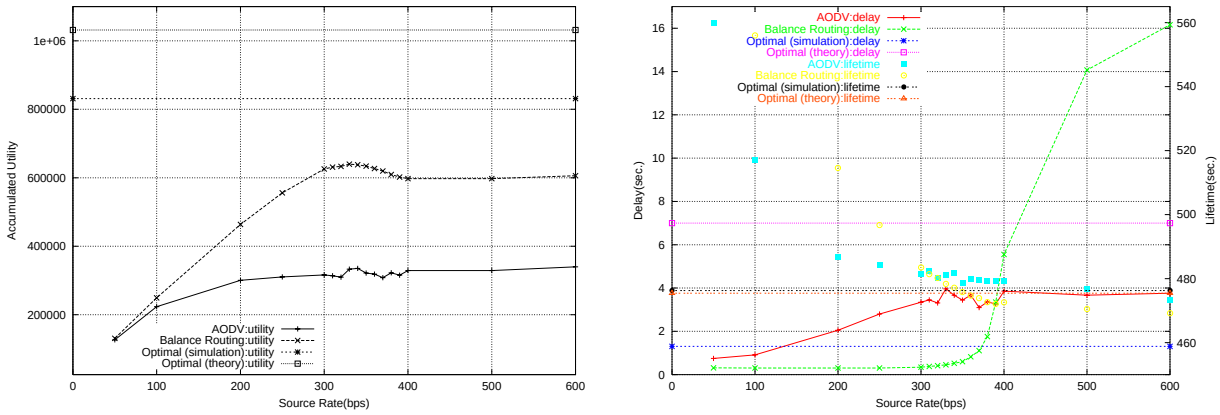
are used for packets that originate from different sensors, giving rise to different source rates for sensors with different values.

The necessity of deriving the node capacity: To investigate whether or not deriving the network capacity, C_i, for each node i is truly necessary, we compare the energy-aware flow control solution obtained by using the network capacity derived in Section 4.3 to that obtained by using arbitrary values of C_i. Fig. 4.8 gives the simulation results.

The scenario with the network capacity $C_i = 2.4$ kbps for all nodes achieves the maximum utility. The result in the accumulated utility deteriorates if either a more or less stringent node capacity constraint is imposed. The solution obtained by using the network capacity derived in Section 4.3 achieves higher utility and incurs a smaller end-to-end latency as well. Without exhaustively testing all the possible values of the network capacity, the conservative network capacity derived in Section 4.3 provides a reasonable setting to the problem.

4.5 Summary

In this chapter, we formulate the problem of data transport in sensor networks as an optimization problem whose objective function is to maximize the amount of information (utility) collected at sinks (subscribers), subject to the flow, energy and channel bandwidth constraints. Also, based on a Markov model extended from [6], we derive the link delay and the node capacity in both



(a) (b)

Figure 4.7: Performance comparison of the energy-aware flow control solution against AODV routing and load balanced routing with respect to (a) utility and (b) end-to-end packet delay and system lifetime under a wide range of source rates. The utility values of packets are uniformly distributed in $[1, 100]$.

the single and multi-hop environments, and figure them in the problem formulation. We study three special cases under the problem formulation. In particular, we consider the energy-aware flow control problem, derive an energy aware flow control solution, and investigate via *ns-2* simulation its performance. The simulation results indicate that the solution to the energy-aware flow control problem achieves highest accumulated utility as compared to AODV and balanced routing with the use of various source rates.

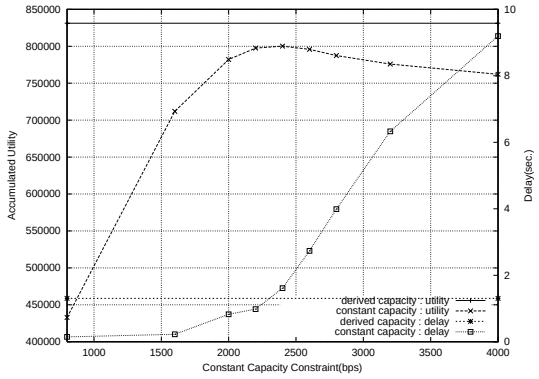


Figure 4.8: Performance comparison between the energy aware flow control solution obtained by using the network capacity derived in Section 4.3 to that obtained by using arbitrarily chosen values.

Chapter 5

A Distributed, Energy-aware, Utility-based Approach for Data Transport

The problem formulation in Chapter 4 (Eqs. (4.1)–(4.4)) is a non-convex programming problem and a centralized approach is used to solve the optimization problem. As the centralized approach cannot quickly adapts to dynamic network changes, in this chapter we devise a distributed, energy-aware, utility-based approach. The key to devising such a distributed solution is to transform the optimization problem to a convex programming problem. That is, the objective function has to be concave and the range of the feasible control variables is within a convex set. The advantage of a convex programming problem is that its dual problem can be solved in a distributed manner, as the control variables in the dual problem can be expressed in a separable form. Furthermore, there is no duality gap between the primal and dual problems.

There exist several challenges that we have to tackle in order to apply utility-based approaches to wireless sensor networks. The contributions of our proposed approach can be summarized in three directions:

(I) Estimating node capacity in wireless multi-hop environments As all the outgoing links from a node share the same channel in wireless multi-hop environments, capacity constraints should be imposed on the node capacity rather than on the link capacity. Constraints on the link capacity suffice only when a node is equipped with multiple transceivers operating at different channels or with directional antennas. Moreover, since the wireless channel is a shared medium, the channel capacity [1] is shared by all devices within the radio transmission range. As a result, different from the wireline environments in which the capacity of a link is given in its specification,

[1] with the unit of bits/sec.

the capacity of a node in wireless environments is no longer a constant but changes in different topologies and traffic status. It is nontrivial to determine the capacity of each node needed in the optimization problem. In this thesis, we propose a capacity estimation method that adapts to the traffic load. The estimation is made based on (i) the measurements of average outgoing throughput and (ii) the feedback information of buffer overflows and unsuccessful transmissions. All the information can be obtained locally and the required computation is not intensive.

(II) Linearizing energy constraints It is necessary to consider energy constraints because unattended sensors are equipped only with limited energy. If the system lifetime is considered as a control variable in the energy constraints such as in Chapter 4, the function that describes the energy constraint includes a product term of the system lifetime and the data rate of sources and is hence a non-convex function. As a non-convex programming problem cannot be solved in a distributed manner, we need to transform the energy constraint to a convex function. One method is to properly set the value of the system lifetime in advance and control the data rate of a node (and hence its total energy consumption rate) so as to sustain its battery lifetime longer than the specified lifetime. In this manner, the energy constraint becomes a linear function, as the system lifetime becomes a parameter but no longer a control variable. The price of the energy constraint is periodically adjusted according to the current energy consumption rate. Furthermore, both the capacity and energy constraints can be quantified in terms of the price based on how tight these constraints are. A more stringent constraint leads to a higher price.

(III) Integrating routing dynamics in the optimization Like most existing flow control approaches, the proposed approach solves the problem of maximizing the amount of information delivered in two phases: a set of routes are determined in the first phase and then a convex programming problem is solved given the set of routes in the second phase. The resulting solution may not be optimal, but it has been shown in [81] that the problem of solving both routing and flow control simultaneously is NP-hard. As such, we incorporate the optimization results into routing, and propose a modified version of Ad hoc On Demand Distance Vector (AODV) routing protocol. The proposed routing protocol provides sensors with opportunities to select a route with a smaller price, and at the same time, improve the overall utility of the system. It is composed of three phases, route initialization (*RINT*), route request (*RREQ*) and route reply (*RREP*). In the first RINT phase, sink-trees constructed by sinks establish active connections from sensors to sinks and reduce the overhead of route discovery. As these sink-tree routes might incur high price due to the

limited energy of certain intermediate nodes and/or congestion along these routes. The RREQ and RREP phases in the proposed protocol are then to find *alternate* routes with smaller costs. One issue has to be considered in particular: suppose a data flow f is to be switched to a new route, then the "losses" of data flows that are originally routed on this new route should be well compensated by the gain in switching the data flow f to this new route. By applying a price estimation method, we can reduce the impact of redirecting a data flow on the flows that are originally routed on the new route. In addition, the overheads incurred in the RREQ and RREP phases are controlled properly not to overload the networks.

The rest of the chapter is organized as follows. We define the problem and present a distributed flow control approach in Section 5.1. Linearization of energy constraints, on-line estimation of the node capacity, and integration of dynamic routing with flow control are treated in Section 5.2, 5.3, and 5.4, respectively. Following that, we present the simulation results in Section 5.5.

5.1 Problem Definition

In this section we briefly describe the scenario of applications in sensor networks to which the proposed utility-based approach is applied. Suppose a volcano observation system is to be implemented with a wireless sensor network. Sensors are placed near the volcanoes to monitor their activities. The data rate of a sensor depends on the information of the activity a sensor observes. For instance, a low data rate is sufficient for the site of a dormant volcano. When a sensor detects an abnormal activity at a dormant volcano, it increases the data rate. If a volcano erupts, sensors then collect and transmit the status data with the highest data rate. The objective of the system is to deliver the largest amount of information (but not the largest amount of raw data) during the period of the system lifetime.

The same assumptions (A1)-(A3) in Chapter 4 are made for the distributed utility-based approach. We formulate the optimization problem as a convex programming problem: to maximize the total utility of data collected at sinks, subject to the channel capacity constraints and the energy constraints. The notations used in this chapter are defined in the following:

- $\mathcal{U}_s(x_s)$: the utility function of the data rate x_s generated from a sensor s and sent to a sink. The utility function is assumed to be a strictly concave function. The range of the source rate x_s needs to be within the range $I_s = [m_s, M_s]$;
- S_n and S_i: the set of sensors and sinks in the sensing field;

- $\mathcal{N}(i)$: the set of sources that use node i as a relay node (including node i itself);
- $\mathcal{P}(s)$: the set of nodes relaying packets for the source s (including source s itself);
- E_i: the amount of energy initially equipped with node i;
- e_i: the energy consumed in the idle state per unit time;
- e_s and e_r: the additional energy consumed in transmitting and receiving one unit of data rate per unit time;
- T_ℓ: the pre-specified, desired system lifetime.
- C_i: the channel capacity of node i;

For ease of description, in this section, we only consider the node capacity constraints in the optimization problem. The energy constrains will be considered in Section 5.2. Similar to the problem considered by Low *et al.*[52] in wireline networks, the optimal flow control problem in wireless networks is to maximize the sum of utility of all sources subject to the node capacity constraint, C_i, at each node i. The primal optimization problem can be expressed as:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{s \in S_n} \mathcal{U}_s(x_s) \\ s.t. \quad & \sum_{s \in \mathcal{N}(i)} x_s \leq C_i \qquad \forall i \in S_n \end{aligned} \tag{5.1}$$

Since the utility function is strictly concave, there exists a unique maximization solution for the primal problem. However, the constraints require the knowledge of all source rates. The key to making the optimization problem separable is to consider the dual problem. First we define the Lagrangian function as:

$$\begin{aligned} L(\mathbf{x}, \mathbf{p}) &= \sum_{s \in S_n} \mathcal{U}_s(x_s) + \sum_{i \in S_n} p_i \cdot (C_i - \sum_{s \in \mathcal{N}(i)} x_s) \\ &= \sum_{s \in S_n} (\mathcal{U}_s(x_s) - x_s \cdot \sum_{i \in \mathcal{P}(s)} p_i) + \sum_{i \in S_n} p_i C_i, \end{aligned} \tag{5.2}$$

where $\mathbf{p} = \{p_i\}$ is the vector of the Lagrangian multipliers, and p_i can be regarded as the capacity price per unit bandwidth usage at node i. Next, the dual problem is defined as

$$\min_{\mathbf{p} \geq \mathbf{0}} D(\mathbf{p}) \tag{5.3}$$

with the objective function:

$$D(\mathbf{p}) = \max L(\mathbf{x}, \mathbf{p}) = \sum_{s \in S_n} \mathcal{B}_s(p^s) + \sum_{i \in S_n} p_i C_i, \tag{5.4}$$

where

$$\mathcal{B}_s(p^s) = \max_{x_s \in I_s} \mathcal{U}_s(x_s) - x_s p^s, \tag{5.5}$$

$$p^s = \sum_{i \in \mathcal{P}(s)} p_i. \tag{5.6}$$

Because the second equality of the Lagrangian function in Eq. (5.2) can be expressed separable in x_s, the optimization of the dual problem can be solved at each node in a distributed manner. By Kuhn-Tucker theorem, the optimality point happens at: $\nabla L(\mathbf{x}, \mathbf{p}) = \mathbf{0}$. Then $x_s(p)$, the unique maximizer of Eq. (5.5), can be obtained:

$$x_s(p) = [\mathcal{U}'^{-1}_s(p^s)]^{M_s}_{m_s}, \tag{5.7}$$

where $[z]^b_a \stackrel{\triangle}{=} \min\{\max\{z, a\}, b\}$ and $\mathcal{U}'^{-1}_s(\cdot)$ is the inverse function of the derivative of utility function $\mathcal{U}_s(\cdot)$. Therefore, source s can determine its optimal rate if the information of p^s in Eq. (5.6), the sum of the price per unit bandwidth at each node on the path for source s, is available. On the other hand, the optimal price $\mathbf{p}$ of solving the dual problem in Eq. (5.3) can be obtained by applying the gradient projection method [4]. The price for node i, p_i, can be adjusted according to:

$$\begin{aligned} p_i(t+1) &= [p_i(t) - \alpha \frac{\partial D(\mathbf{p}(t))}{\partial p_i}]^+ \\ &= [p_i(t) - \alpha(C_i - x^i(\mathbf{p}(t))]^+, \end{aligned} \tag{5.8}$$

where $[z]^+ \stackrel{\triangle}{=} \max\{z, 0\}$, α is a small positive step size and $x^i(\mathbf{p}) = \sum_{s \in \mathcal{N}(i)} x_s(\mathbf{p})$ is the sum of the rates of all flows passing through node i at time t. Finally, the solution of the optimal rate and price can be solved iteratively in Eqs. (5.7) and (5.8) at sources and routers, respectively. Low *et al.* [52] show that the optimal solution converges provided that the step size α is sufficiently small.

Price Information Dissemination To determine the optimal source rate x_s, each source s needs to know the total price p^s on the path to the destination. A simple implementation method is to carry the price information of the path in the data packet, update it accumulatively from

the source to the destination, and enable the destination to return the total price to the source in an explicit transport-level acknowledgment (ACK) packet. In spite of its simplicity, this method is not well-suited for resource-limited sensor networks since the overheads of explicit ACKs are considerable and reliable and timely delivery of ACK packets must be ensured.

In our proposal, the price information is conveyed through link-level ACKs in 802.11-like MAC protocols. When an intermediate relaying node receives a data packet, it retrieves the accumulative price of this flow from its flow information table [2] , and attaches the price to the link-level ACK packet. The accumulative price includes the price at the relaying node and all its downstream nodes (i.e., the nodes on the path towards the destination). Upon receipt of an ACK packet carrying the price information, the upstream node updates the price of this flow in its table. The price information of the entire path is eventually propagated from the last hop of the path to the source. By carrying the price information in the ACK packet only when the price of the flow changes, the overheads can be further reduced. A bit in the ACK packet header can be used to indicate whether the price information is included.

The price adjustment procedure in Eq. (5.8) executed at a node requires the sum of the rates of all the outgoing flows. This can be measured locally without incurring extra communication overheads. One drawback of the proposed approach is that each intermediate node needs to retain per-flow information. However, as sensor data are usually aggregated and processed at cluster heads, the number of data flows transported in the network is at most comparable to the number of cluster heads and hence is not prohibitively high.

5.2 Linearization of Energy Constraints

How to reduce energy consumption is an important issue for battery-powered sensors and hence it is necessary to figure in energy constraints in the optimization problem. Obviously there exists a trade-off between maximizing the amount of information delivered and prolonging the system lifetime. It would be desirable to increase data rates of certain flows only when certain events of interest take place in the proximity of the sources. However, it is difficult to predict when events of interest will take place in the future. Therefore, a reasonable energy constraint is to preserve energy to ensure the system remains operational at least beyond a pre-specified system lifetime. In this section, we elaborate on how to figure in such a linear energy constraint in Eq. (5.1) so as to

[2] In the case that an intermediate relaying node always forwards packets to the same next hop towards the destination, all flows passing through it have the same path price and thus no flow table is required.

make the optimization problem separable.

Let t be the current time and $E_i(t)$ be the current remaining energy of node i. For a sensor node to operate beyond a pre-specified time instant T_ℓ, the current data rate should be determined, so that the energy incurred in the transmission, reception, and idle states for the remaining interval till time T_l is smaller than the current remaining energy $E_i(t)$. Specifically,

$$((e_s + e_r) \cdot \sum_{s \in \mathcal{N}(i)} x_s - e_r x_i + e_i) \cdot (T_l - t) \leq E_i(t), \quad \forall i \in S_n. \tag{5.9}$$

In other word, each node needs to control the data rate (that includes traffic originating from the node and transit traffic), such that its energy consumption rate is below the rate that sustains the specified system lifetime. We call this rate the *intended energy consumption rate*, and denote it as

$$b_i(t) \stackrel{\triangle}{=} E_i(t)/(T_l - t). \tag{5.10}$$

The optimization problem that includes both the node capacity and linear energy constraints is

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{s \in S_n} \mathcal{U}_s(x_s) \\ s.t. \quad & \sum_{s \in \mathcal{N}(i)} x_s \leq C_i, & \forall i \in S_n \\ & (e_s + e_r) \cdot \sum_{s \in \mathcal{N}(i)} x_s - e_r x_i + e_i \leq b_i(t) & \forall i \in S_n \end{aligned} \tag{5.11}$$

Note that the energy constraint is a linear function because the *intended energy consumption rate*, $b_i(t)$ of a node i is now a parameter. Therefore, the optimization problem in Eq. (5.11) is a convex programming problem and can be solved in a distributed manner. Similar to the derivation in Section 5.1, the Lagrangian function is now defined as

$$\begin{aligned} & L(\mathbf{x}, \mathbf{p}) = \sum_{s \in S_n} \mathcal{U}_s(x_s) + \sum_{i \in S_n} \{p_{c_i} \cdot (C_i - \sum_{s \in \mathcal{N}(i)} x_s) + \\ & p_{l_i} \cdot [b_i(t) - ((e_s + e_r) \cdot \sum_{s \in \mathcal{N}(i)} x_s - e_r x_i + e_i)]\} = \\ & \sum_{s \in S_n} \{\mathcal{U}_s(x_s) - x_s \cdot [\sum_{i \in \mathcal{P}(s)} (p_{c_i} + p_{l_i}(e_s + e_r)) - p_{l_s} e_r]\} \\ & + \sum_{i \in S_n} (p_{c_i} C_i + p_{l_i}(b_i(t) - e_i)), \end{aligned} \tag{5.12}$$

where p_{c_i} and p_{l_i} are the price charged for capacity constraint and lifetime constraint at node i, respectively. Let the accumulative prices along the path from source s to the destination for

the capacity and lifetime constraints be denoted as $p_c^s \stackrel{\triangle}{=} \sum_{i \in \mathcal{P}(s)} p_{c_i}$ and $p_l^s \stackrel{\triangle}{=} \sum_{i \in \mathcal{P}(s)} p_{l_i}$. Similar to Eq. (5.7), each source s, controls its optimal rate based on the prices for the capacity and lifetime constraints:

$$x_s(p) = [\mathcal{U}_s'^{-1}(p_c^s + p_l^s(e_s + e_r) - p_{l_s} e_r)]_{m_s}^{M_s}. \tag{5.13}$$

Besides, the prices for the capacity and lifetime constraints are adapted at each node periodically based on the usage of the resource:

$$\begin{cases} p_{c_i}(t+1) = [p_{c_i}(t) - \alpha(C_i - x^i(\mathbf{p}(t))]^+, \\ p_{l_i}(t+1) = [p_{l_i}(t) - \beta[b_i(t) - ((e_s + e_r)x^i(\mathbf{p}(t)) \\ \qquad -e_r x_i(\mathbf{p}(t)) + e_i)]]^+, \end{cases} \tag{5.14}$$

where α and β are small positive step sizes for adjusting the capacity and lifetime prices, respectively. Even though $b_i(t)$ might change with time, the price for the energy constraint is only adjusted periodically according to the gap between the current energy consumption rate and the current *intended energy consumption rate* $b_i(t)$. When the current energy consumption rate exceeds $b_i(t)$, the price for the energy constraint is increased and vice versa. Similarly, the source data rate and the price are adjusted iteratively at each node with the use of Eqs. (5.13) and (5.14), respectively.

The above approach combines both the prices for the capacity and energy constraints into a single entity. The importance of both capacity and energy constraints can be quantified in terms of the prices, based on how tight the constraints are. A more stringent constraint leads to a higher price.

5.3 Estimation of Node Capacity

One major challenge of employing utility-based optimization (Eq. (5.1)) in the multi-hop wireless environment is how to estimate the node capacity, C_i for each node i. In contrast to the link capacity in wireline networks which is given in its link specification, the node capacity in the multi-hop wireless environment is no longer a constant but highly dependent upon the nodal distribution in its neighborhood and the traffic conditions at other neighboring nodes. In Chapter 4, a static node capacity is derived based on the conservative assumption that all the two-hop neighboring nodes are backlogged. The node capacity thus derived might be conservative when some of neighboring nodes are not backlogged.

In this thesis, we take the derived node capacity, $\widehat{C}_i$, in Chapter 4 as the *nominal* capacity. In addition, we propose a light-weight capacity estimation method that adapts to the traffic conditions and buffer status. The final estimate of the node capacity is a weighted sum of the nominal capacity, $\widehat{C}_i$, and the dynamically estimated capacity, $\overline{C}_i$, i.e.,

$$C_i = \lambda \widehat{C}_i + (1 - \lambda)\overline{C}_i. \tag{5.15}$$

The on-line estimated node capacity, $\overline{C}_i$, is calculated as the average throughput attained by all the outgoing flows, i.e.,

$$\overline{C}_i = \kappa \cdot \overline{r_i}, \tag{5.16}$$

where $\overline{r_i}$ is the averaged, aggregated throughput attained by all the outgoing flows, and κ ($\kappa < 1$) is a parameter that ensures system stability. As the queuing size grows indefinitely when the incoming rate is greater than or equal to the serving capacity, the parameter κ is used to prevent the system queue from growing unbounded. (We will discuss below how to on-line tune κ.) The average throughput attained by all the outgoing flows, $\overline{r_i}$, is measured over the past T_c seconds as follows. Each node records the delay, d_j, incurred by each outgoing packet P_j with packet size, $Size_j$, to the next hop, i.e. the latency from the time of retrieving the packet from the head of queue to the time of receiving the corresponding ACK packet. The average throughput attained during the past T_c seconds is calculated as

$$\overline{r_i} = \frac{1}{P} \sum_{j=1}^{P} \frac{Size_j}{d_j}, \tag{5.17}$$

where P is the number of packets transmitted in T_c. A low-pass filter with an exponentially weighted moving average can be used to filter out transient fluctuation of the calculated attainable throughput.

We use a simple feedback control mechanism to on-line adjust κ. The value of κ is increased by a small value, $\delta\kappa_+$, if the following two conditions hold: (i) During the past T_c seconds, no packet drops as a result of buffer overflow or failure to reach the next hop (due to contention); and (ii) the number of packets currently in the buffer is less than a pre-determined threshold, Q_{low}. On the other hand, the value of κ is decreased by a small value, $\delta\kappa_-$, if more than N_p packets are dropped due to either buffer overflow or failure to reach the next hop in the past T_c seconds. For the other cases, κ remains the same value.

5.4 Integrating Routing Dynamics in the Optimization Problem

The utility optimization problem considered in Section 5.2 is solved in two phases. In the first phase, the routes for all the sources are determined based on hop counts. Then in the second phase convex programming is solved given the set of the given routes. The solution based on fixed routes is not optimal, but it has been shown in [81] that the optimization problem considering both routing and flow control decisions simultaneously ($\max_{\mathcal{R}} \max_{\mathbf{x}} \sum_{s \in S_n} \mathcal{U}_s(x_s)$) is NP-hard.

Instead of searching the entire space, we propose to incorporate the optimization results into routing, and propose a modified version of Ad hoc On Demand Distance Vector (AODV) routing protocol. The proposed routing protocol provides sensors with opportunities to select routes with smaller prices and improve the overall utility of the system. The objective of integrate dynamic routing to the utility optimization problem is to increase the overall utility while keeping the system stable. Route changes should be much less frequent than rate changes in flow control in order to maintain the system stability. The trade-off between maximizing the utility and keeping the route stability is studied in [81]. The proposed routing protocol attempts to balance the trade-off between maximizing the utility and reducing routing overheads, and is composed of three phases: route initialization (*RINT*), route request (*RREQ*) and route reply (*RREP*). In what follows, we elaborate on each phase and highlight the differences between AODV and the proposed protocol. Fig. 5.1 summarizes the three phases.

(1) Route Initialization (RINT) In the first *RINT* phase, all the sinks construct sink-tree routes to the sensors. Unlike generic ad hoc networks, the destinations for all sensors in sensor networks are usually the sinks. In addition, sensors are usually of low mobility. Therefore, these sink-tree routes establish connections proactively between sources and destinations, and reduce the overheads incurred in route discovery.

The operations of constructing sink-trees which originate from different sinks are as follows: At the beginning of system operation, all the sinks broadcast *route initialization (RINT)* packets. A *RINT* packet carries the information of the sink ID, the number of hops traversed so far, and the accumulative energy on the path.[3] Upon receipt of a *RINT* packet, a node creates an entry in its routing table (if an entry indexed by the ID of the originating sink does not exist), storing the information of the sink ID, the ID of the node from which the *RINT* packet arrives (i.e., the next hop to the sink), the hop count, and the accumulative prices to the sink [4]. The node only

[3] Another choice is the minimum energy of a node along the path.
[4] The initial price is the product of a default value and hop count

forwards the *RINT* packet under the two conditions: (i) if the hop count recorded in the *RINT* packet received is less than that kept in the routing table entry (indexed by the sink ID); or (ii) if the *RINT* packet originates from a new sink and the hop count recorded in the packet is at most δh more than the hop count to the closest sink. The value[5] of δh determines the size of the overlap regions of different sink-trees. Before broadcasting a *RINT* packet, the forwarder increases the hop count in the packet by one and sets a back-off timer to reduce the possibility of colliding with *RINT* packets from other forwarder nodes. The value of back-off timer consists of a deterministic part (that is proportional to the traversed hop count h) and a uniform random part:

$$D = c_1 \cdot h + U(c_2), \tag{5.18}$$

where both c_1 and c_2 are system parameters. Setting of the back-off timer in this manner makes a node forward a *RINT* packet that traverses less hops earlier, and all the sink trees (originating from different sinks) grow approximately at the same rate. A *RINT* packet stops to be forwarded (and hence a sink tree stops growing) when the packet arrives in the proximity of the boundary of the "territories" of two sink nodes. A node in the overlap area has the flexibility to select any sink as its destination.

After a source chooses a sink with the least hop count as its initial destination, it determines the next-hop node as the one with the least hop count to the destination. In the case of a tie, the node with the most energy is chosen as the next-hop node. A similar forwarding rule is also applied to relay nodes. When a node receives the first packet from a new source, it relays the packet to the next hop with the least hop count to the destination. If there exist multiple next-hop candidates, the next-hop node is chosen in round robin fashion for load balancing. The per flow (source) information is also needed to be kept in intermediate routers because in the flow optimization problem the price information from downstream nodes needs to be properly propagated to the source. In addition to the flow (source) ID, the sink ID, the hop count, the next-hop ID, and the accumulative energy on the path, the routing table of a node also contains the cumulative prices for the capacity and energy constraints to the sink. Hence, at the end of the *RINT* phase, each source or intermediate node can select (based on hop count and flow count information) the next hop for the flow that originates from itself or transit flows, respectively and the flow control algorithm is executed given the routes established in the first phase.

[5] $\delta h = 1$ in this thesis to ensure routes are loop-free.

(2) Route Request (RREQ) The routes established in the *RINT* phase may incur high prices due to the limited energy of certain intermediate node(s) or congestion along the routes. Different from route discovery in AODV, the *RREQ/RREP* phases aim to find alternate routes with smaller costs (prices) under the two conditions: (i) the overheads incurred in the *RREQ/RREP* phases are kept minimal; and (ii) in the case that a data flow f is to be switched to a new route, the "losses" of data flows that are originally routed on this new route should be well compensated by the gain in switching the data flow f to this new route.

In order not to overload the network, the restrictions on the usage of *RREQ* packets are imposed on both the sources and intermediate nodes. A source is allowed to search for an alternate route only when it detects that an event leads to high utility but the price of using the current path is so high that the data rate falls much below the possible maximal rate M_s. The original route remains operational until an alternate route with a cheaper price is found. On the other hand, selection of the next hop for either the source or relay nodes is restricted by both the hop count and the price constraints. The source sets a upper bound on the hop count to be equal to the sum of δh and the original hop count to its destination. Besides, the price of the new route must be cheaper than the original price by a certain level; otherwise, it may not be worthwhile to distribute the *RREQ* packet. (The details on how to determine whether or not the price along an alternate path is sufficiently cheap will be given below.) Furthermore, to prevent excessive exchanges of *RREQ* and *RREP* packets, *RREQ* packets are sent by unicast rather than broadcast. The node with the cheapest price in the routing table is chosen as the candidate of the next hop. A *RREQ* packet contains the source ID, the original sink ID, the upper bound on the hop count, the original price, the value of the event detected at the source, and finally the accumulative prices, $p_{c_{up}}$ and $p_{l_{up}}$, for both the capacity and lifetime constraints of all upstream nodes (including the source) on the new route.

Recall that the purpose of changing the route is to increase the *overall* utility of the system. As such, we need to consider not only the utility gain of the flow that is be switched to a new route, but also the utility loss of all the flows that are originally routed on the new route (due to the increased traffic from the redirected flow). The criterion for accepting a route change is as follows: Suppose a source s changes its route from path $\mathcal{P}_o(s)$ to path $\mathcal{P}_n(s)$. The route change leads to the utility changes of all the flows using node s or nodes on $\mathcal{P}_o(s)$ or $\mathcal{P}_n(s)$. In order to increase the

system utility, the change in the utility of the overall system

$$\sum_{i:\mathcal{P}_o(i)\cap\{s,\mathcal{P}_o(s),\mathcal{P}_n(s)\}\neq\emptyset} (\mathcal{U}_i(x_{i_n}) - \mathcal{U}_i(x_{i_o})) \tag{5.19}$$

should be positive, where x_{i_n} and x_{i_o} represent the new and old rate of node i. The flows using $\mathcal{P}_o(s)$ benefit from the reduced traffic and thus

$$\sum_{i:\mathcal{P}_o(i)\cap\{\mathcal{P}_o(s)\}\neq\emptyset} (\mathcal{U}_i(x_{i_n}) - \mathcal{U}_i(x_{i_o})) \geq 0. \tag{5.20}$$

As a result, the following inequality holds:

$$Eq.\ (5.19) \geq \sum_{i:\mathcal{P}_o(i)\cap\{s,\mathcal{P}_n(s)\}\neq\emptyset} (\mathcal{U}_i(x_{i_n}) - \mathcal{U}_i(x_{i_o})). \tag{5.21}$$

Therefore, as long as the utility change of all the flows using node s or nodes on $\mathcal{P}_n(s)$ is greater than zero, the change in the utility of the overall system is positive. Testing of the utility change is performed at all the nodes that receive RREQ packets. The challenge is how to estimate the new rate x_{i_n} after the route change. The approximation we use is that the intermediate forwarder node f determines its own price change and the new rate of all flows passing through itself in the following six steps:[6]

(S1) Node f retrieves the cheapest price towards any sink satisfying the hop count constraint. Since this price is calculated without consideration of the redirected traffic, we denote it as the *priori* price, p^f_{pri}.

(S2) Node f estimates the data rate of source s based on the value of the event (carried in the *RREQ* packet) and the sum of the upstream price and the priori price[7]:

$$x_{s_{pri}} = [\mathcal{U}'^{-1}_s(p_{up} + p^f_{pri})]^{M_s}_{m_s}. \tag{5.22}$$

(S3) According to the increased traffic from the redirected flow, node f calculates its *posterori* price, p^f_{post}, by Eq. (5.14).

(S4) Similar to (S2), the *posteriori* data rate of source s is determined based on the price $p_{up}+p^f_{post}$.

[6]For simplicity, only the capacity price is presented here.
[7]we assume the forwarder knows the function of $\mathcal{U}_s(\cdot)$ as long as the value of the source is known.

(S5) Similar to (S1)-(S4), for each flow from node i passing through node f, node f first determines the original price of node i. Next the price increase of node f is added to node i's price. Finally the new rate is estimated based on the new price.

(S6) Node f calculates the utility change given in Eq. (5.21) and determines whether to forward the *RREQ* packet based on the following criterion:

$$\mathcal{U}_s(x_{s_n}) - \mathcal{U}_s(x_{s_o}) \geq c_3 \cdot [\sum_{i \in \mathcal{N}(f)} (\mathcal{U}_i(x_{i_o}) - \mathcal{U}_i(x_{i_n}))]. \tag{5.23}$$

Only when the gain in the utility increase of source s (the term on the left hand side in Eq. (5.23)) is greater than or equal to c_3 ($c_3 > 1$) times of the utility decrease in all existing flows, the RREQ packet is forwarded.

By applying a price estimation method, we can reduce the impact of redirecting a data flow on the flows that are originally routed on the new route. Finally, the forwarding process continues until the *RREQ* packet reaches to the last hop on the new path, at which point the *RREP* phase commences.

(3) Route Reply (RREP) When a *RREQ* packet reaches the last hop, the node next to the sink sends back a *RREP* packet with the price of the new route equal to its own price[8]. Similar to the price estimation method (S1)-(S4) in the *RREQ* phase, the posterior price is calculated by figuring in the traffic increase due to the redirected flow. Furthermore, the sum of the utility loss of all flows passing through each node on the new route is calculated (following (S5)-(S6)). Both the posterior downstream price and the utility loss are carried in the *RREP* packet. All the intermediate relay nodes follow the same procedures to calculate the downstream price and accumulative utility loss. When the source receives the *RREP* packet, it determines whether to change its route based on Eq. (5.23).

5.5 Simulation Results

We evaluate the performance of the distributed utility-based approaches using the *ns-2* [98] simulator. A total of 60 sensors and 4 sinks are deployed in a square grid given in Fig. 5.2. The distance between neighboring sensors is 200 meters. We assume the radio transmission range is

[8]We assume the sinks have unlimited resource in both outgoing capacity and energy resource and hence the price of sinks is equal to zero.

Phase I: Route Initialization

Sink-tree construction:
1. Sinks: broadcast a *RINT* packet
2. Forwarders: upon receipt of a *RINT* packet:
3. if (RINT packet satisfies the hop count constraint)
4. hop count++
5. set a back-off timer based on the hop count
6. broadcast the *RINT* packet

Selection of the initial route:
1. Sources: choose the destination and the next hop based on the hop count information
2. Forwarders: choose the next hop based on the hop count and flow count information

Phase II: Route Request

Sources:
1. if (an event with high value && too expensive price)
2. unicast *RREQ* to the node with the cheapest price

Forwarder f: upon receipt of a *RREQ* packet:
3. follow (S1)-(S6) to calculate the posterior price and utility change
4. if (utility change < 0 || f is the last hop)
5. sends back *RREP*
6. else
7. forwards *RREQ* to the node with the cheapest price

Phase II: Route Reply

Forwarders: upon receipt of a *RREP* packet:
1. follow (S1)-(S6) to calculate the posterior price and utility change and append both to the RREP

Source: upon receipt of a *RREP* packet:
2. if (cheaper price && utility change > 0)
3. redirects the flow to the new route
4. else
5. stays in the original route
6. sends another *RREQ* if a potential good route exists

Figure 5.1: Three phases of the proposed dynamic routing algorithm.

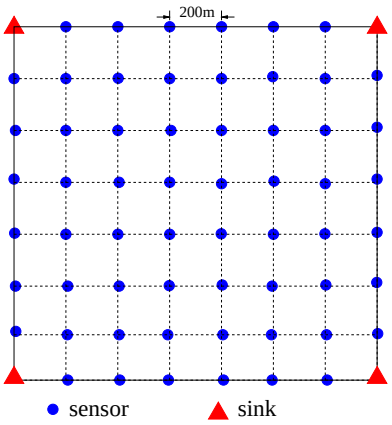


Figure 5.2: The topology used in the simulation study. A total of 4 sinks and 60 sensors are deployed on a square grid. The distance between neighboring sensors is 200 meters, and each sensor has at most 4 neighbors.

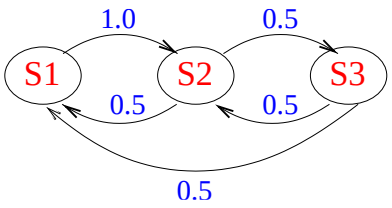


Figure 5.3: Markov state model of value of packets at a sensor.

250 meters. Therefore, each sensor has at most 4 neighbors. The data transmission rate of the wireless channel is assumed to be 40 kbps. The utility function used in the simulation is defined as $\mathcal{U}_s(x_s) = v_s \cdot log(x_s + 1)$, where v_s and x_s are the utility value of a packet and the source rate (in units of #packets/second) for sensor node s, respectively. The function $\mathcal{U}_s$ is a non-decreasing and concave function of node s's sending rate. To test the performance under dynamic environments, the values of packets are not constant, but are specified according to Figure 5.3 and Table 5.1. We envision a volcano monitoring system used to record the volcano activities. When a volcano stays at the dormant status (state 1), sensors transmits data at a low rate to sinks. If sensors detect abnormal events (state 2), a higher data rate is required to facilitate transport of data back to the sink for further analysis. Once a volcano eruption event is likely to take place (state 3), the transmission rate should be further raised.

We use a Markov chain model given in Fig 5.3 to generate the value of packets captured at each sensor. The parameters used in each of the Markov states are listed in Table 5.1. The time period of a state is uniformly random distributed in [10, Max. Period] seconds. The parameters for power consumption follows the setting in [12], i.e., the power consumption incurred in the transmission,

State	Max. Period	Value	$M_s(bps)$	$m_s(bps)$
S1	200 seconds	1	100	25
S2	50 seconds	10	500	25
S3	100 seconds.	100	3000	25

Table 5.1: Parameters of the Markov state model.

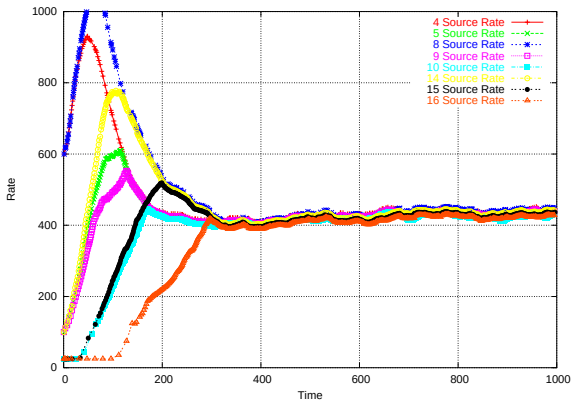


Figure 5.4: Data rates of the eight sensors in the upper left quarter of the 36-node square grid.

reception, and idle state is 1.4, 1.0, and 0.83 W, respectively. Hence, e_i is 0.83 W. e_s and e_r are the additional energy consumed (in addition to e_i) in sending and receiving a bit of data, and are equal to the product of T_b, the time to send a bit and 0.57(=1.4-0.83) and 0.17(=1.0-0.83), respectively. (Note that the units of e_s and e_r are joules per bit.) The payload size of a packet is set to be 70 bytes (including 20 bytes of IP header but not MAC and PHY headers).

System stability in the case that node capacities are on-line measured Before evaluating the utility-based approach in dynamic environments, we first verify whether or not the data rates of sources converge under the case when the node capacites are on-line estimated. In the first set of simulations, a total of 32 sensors and 4 sinks are deployed in a square grid in the same manner as Fig. 5.2, except that each row or column has 6 nodes. (The four sinks located at the corners are labeled as 0, 1, 2, and 3. The 32 sensors are labeled as 4, 5, ..., 35 in sequence, from the top row to the bottom row, and from left to right.) The value of the data from all sensors is fixed to be one, all sensors are initially equipped with 1000 joules (E_i), and the node capacity of each node is on-line measured and calculated. Fig. 5.4 gives the results of data rates from eight sensors located in one quarter of the square grid. At the beginning of the simulation (0-100 seconds), only sensor nodes 4 and 8 — the two sensors next to the sink – have high data rates because they do not incur prices from the other sensors. All other sensors have low rates due to their high default prices. After the time instant 300 seconds, the data rates of all the sensors become stable at approximately 400 bps

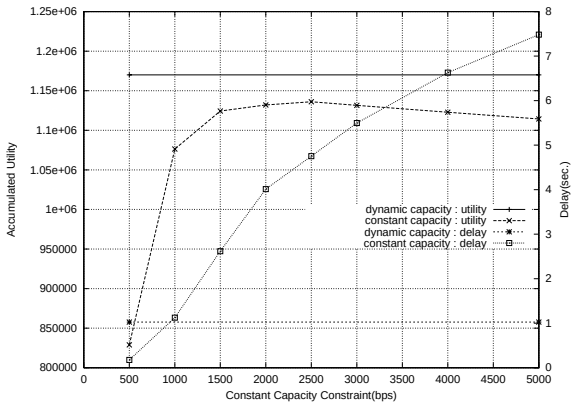


Figure 5.5: The performance of the proposed utility-based approach, when the node capacity is on-line estimated and when it is fixed with 10 different values (labeled in the x-axis).

till the end of simulation.

Advantages of on-line capacity estimation In the second set of simulations, we compare the performance of the proposed utility-based approach with and without on-line nodal capacity estimation enabled. The sensor network given in Fig. 5.2 is used. The value of the data changes according to the Markov model given in Fig. 5.3. All the 60 sensors are initially equipped with 1000 joules. Figure 5.5 gives the result of the proposed approach, when the node capacity is on-line estimated and when it is fixed with 10 different values. The average delay of the proposed approach with fixed capacity values increases as the capacity values increase, while the utility is maximized at the capacity value of approximately 1400 bps. The utility achieved by the proposed approach with on-line estimated node capacities is higher, while the end-to-end latency incurred is smaller. This demonstrates the advantages of on-line estimating the node capacity.

Effects of adjusting lifetime prices The effect of how to adjust the lifetime price (Section 5.2) on the utility and the system lifetime is investigated in the next set of simulations. Initially each sensor is equipped with the same energy, 1000 joules. The intended system lifetime is set to 1050 seconds. (Note that the maximum achievable system lifetime is approximately $1000/0.83 \approx 1200$ seconds, where 0.83 watt is the power consumed in the idle state.) Fig. 5.6 gives the performance comparison of approaches with and without lifetime price adjustment. In particular, we vary two parameters in the approach: the step size used to adjust the lifetime price, β (Eq. (5.14)) and the minimum adjustment period. As expected, the proposed utility-based approach without the energy constraints achieves the highest utility, but the system lifetime is also much shorter. The trade-off between the utility and the system lifetime is also observed in the selection of the step size used to

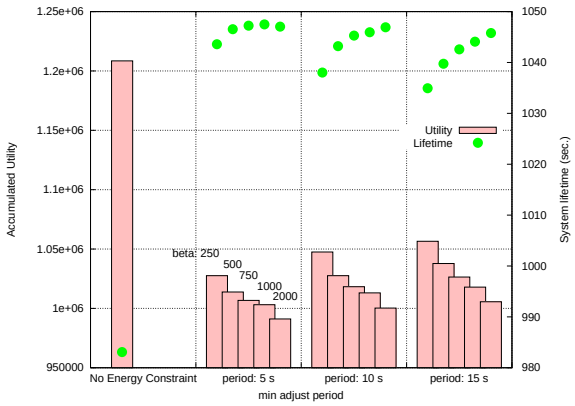


Figure 5.6: The utility and system lifetime for the proposed approaches with and without the lifetime price. Different values of the step size, β, and the adjustment period are considered.

adjust the lifetime price, β (Eq. (5.14)) and the minimum adjustment period. Larger values of β or shorter adjustment periods increase the impact of the lifetime price on the system performance and thus lead to longer system lifetime but less utility.

Results of dynamic routing In this set of simulations, we evaluate the performance of the proposed approach with dynamic routing proposed in Section 5.4. One difference in the set of simulations is that instead of having the same amount of battery capacity, each node is equipped with a battery capacity uniformly distributed in the range of [1000, 2000] joules. Setting different battery capacities for different nodes better simulates deployment of sensors in realistic environments. The intended system lifetime is set to 1200 seconds. Fig. 5.7 gives the results of three different versions of the proposed approaches: (i) consideration of only the capacity constraint, (ii) consideration of both the capacity and energy constraints, and (iii) consideration of both the two constraints and dynamic routing. As compared with the approach with static routing, the approach with dynamic routing performs better both in terms of the utility and the lifetime, although the performance improvement is not dramatic. This is perhaps due to the fact that the energy consumed in overhearing is not considered in the energy constraint of the formulated problem. Although the approach with dynamic routing selects an alternate route to avoid use of nodes with low battery levels, the new route may still be within the carrier sense range of the nodes with low battery levels. In this case, the system lifetime can not be significantly increased.

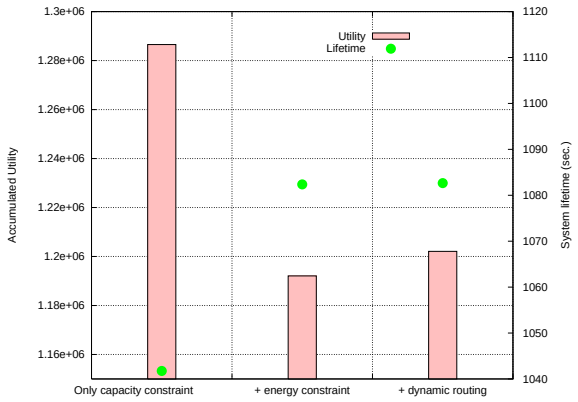


Figure 5.7: Performance comparison of three different versions of the proposed utility-based approaches (from left to right): (i) consideration of only the capacity constraint, (ii) consideration of both the capacity and energy constraints, and (iii) consideration of both the two constraints and dynamic routing.

5.6 Summary

In this chapter we propose a distributed, energy-aware, utility based approach to improve the system utility of wireless sensor networks, subject to both the channel capacity and energy constraints. Our major contributions are (i) on-line estimation of node capacity (to be used in the capacity constraint of the optimization problem) in the multi-hop wireless environment, (ii) inclusion (and linearization) of energy constraints that relate the system lifetime to the data rate, and (iii) incorporation of optimization results in selecting routes that maximize the utility. The simulation results indicate that the utility-based approaches balance between system utility and system lifetime, and can serve as an effective resource utilization mechanism for different types of wireless sensor networks.

Chapter 6

Implementation and Empirical Study on Motes

To validate the design and to empirically study the performance of the proposed work, we implement in this chapter a subset of proposed work on the Mica Motes testbed [18] developed at U.C. Berkeley. We first give a brief introduction of Motes and its operation system, TinyOS in Section 6.1, and present an implementation of the acoustic localization system in Section 6.2. Both delay-based [82] and energy-based localization methods are implemented. Finally, we lay a software architecture fore implementing utility-based data transport in TinyOS in Section 6.3.

6.1 Motes and TinyOS

The Motes architecture was designed with the following characteristics [33]: (i) the devices are of small physical size and incur low power consumption; (ii) as the devices are either collecting information from the sensors or forwarding the data from another device, there is no need for large amounts of buffer; (iii) the capabilities of the controller and the sophistication of the processor are much lower than conventional systems; and (iv) the devices only carry the needed hardware and calls for very modular software. As specified in [18], each Mote is equipped with an Atmel ATMEGA103L processor (8 MHz), a re-programmable co-processor, a short range radio for wireless communication, LEDs (used to display digital values or status of the device), microphones, FLASH memory, and photo/temperature sensors (that can sense temperature and light). The Atmel processor has a serial peripheral interface (Universal Asynchronous Receiver/Transmitter (UART)) for communication between Motes and external peripheral devices. The processor also contains a set of timers and counters that can be used to generate periodic events. A TinyOS operating system [18] runs on top of the processor mentioned above to manage the hardware capabilities of Motes and also to

support the key characteristics for sensor networks. In particular, it has a small footprint and low system overhead, and avoids the problems that arise in a traditional operating system, such as large memory and storage requirements, complex IO systems, high system overhead and complex power-consuming hardware support. This is achieved by not having: (i) a kernel (instead hardware is directly manipulated); (ii) process management (only one process is on the at any time); and (iii) virtual memory (only a single linear physical address space is provided). TinyOS is built upon an event-based model, i.e., as the model helps save power by eliminating the concept of polling or blocking. The device becomes active only if an event is generated. TinyOS consists of a task scheduler and components, where a component is an abstraction of a hardware functional unit (e.g., RFM radio component), synthetic hardware (e.g., radio byte) or high-level software (e.g., a messaging module).

6.1.1 nesC and TinyOS programming

The current TinyOS (version 1.1) uses the *nesC* language [25], an extension to C language to realize the concept of component-based programming. The main advantage of component based programming is that it separates the components composition and configuration. That is, various applications can be easily constructed by changing the configurations between components. The most essential features of *nesC* language are summarized below:

1. Bidirectional interface specifying the relationship between components: The association between different components is specified by the bi-directional interface. Especially, two kinds of functions included in the interface specify the contract between a service user and a service provider. The component of the service user *uses* the interface by calling a non-blocking *command* function implemented in the component of service provider which *provides* the interface. On the other hand, the completion of the command function is signaled by the component of service provider to the service user through a callback function, i.e. the *event* function, implemented in the service user. For instance, the interface *SendMsg* includes a command function **send** and an event function **sendDone**. A component using the interface *SendMsg* needs to provide the implementation of the event function **sendDone**. An interface is declared as a *parameterized interface* when the *interface-parameters* are present (e.g. **interface SendMsg[uint8_t id]**). With the interface-parameters, a single interface can provides multiple interface instances for different types of messages.

2. Flexible component configuration: Components are statically bound via interfaces in the configuration file. Both interface and component can be replaced by other interfaces or components which provide the similar behavior via a alias. For example, **components X as Y** means X is the real component and Y is the alias used in the configuration file. We can easily replace the component **X** by another component, say **Z** in the configuration file.

3. Different scheduling priority for tasks and interrupt handler: TinyOS provides a two-level scheduling hierarchy consisting of tasks and hardware event handlers. Synchronous Code (SC) including functions, commands, events and tasks can be preempted by a hardware interrupt handler (Asynchronous Code (AC)) but can not interrupt with each other. In other word, SC is executed based on *run-to-completion* manner if no hardware interrupts are present. Commands or events which can safely be executed by interrupt handlers must be explicitly marked with the **async** keyword. Although non-preemption characteristic avoids data races among synchronous codes, there still exists potential data races between AC and AC, as well as between AC and SC. In general, a variable, which is reachable from AC and appears at other AC or SC, creates potential data races. *nesC* compiler can detect such data races and give warnings. Such potential data races can be eliminated by using the keyword **atomic** to protect the shared variables from multiple accesses. The code within the atomic brace can not be preempted by other code and is executed "as-if" no other computation occurred simultaneously.

6.2 Empirical Study of Acoustic Target Tracking System on Motes

In acoustic tracking, two most common methods for target localization are the time delay-based [82] and energy-based [46, 73]. In the time delay-based approach, each sensor measures the on-set time of a received signal, where the on-set time is defined as the time instant when the received signal exceeds certain pre-determined threshold. A CH collects from each sensor the relevant timing information for the same signal and determines the difference in the on-set times between different pairs of sensors. Since the time difference is proportional to the distance from the target to a sensor, one can estimate the target location with the use of the multilateration technique in [68]. The challenges of the time delay-based approach are two-fold. First, clocks in all the sensors have to be accurately synchronized. Second, determination of the exact on-set time of a received signal

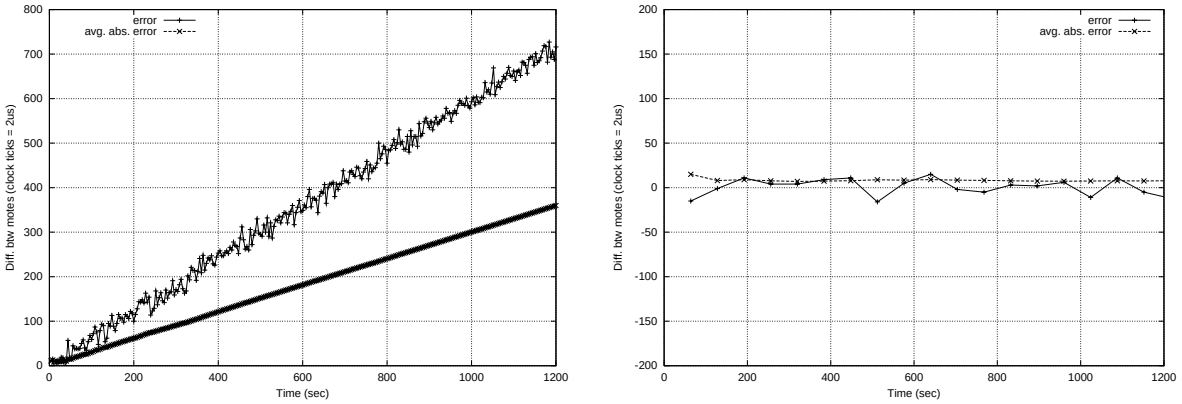
is not an easy task. The localization result is highly susceptible to the estimation error of the on-set time and the effect of echoes. The main application of delay-based localization is to locate the impulsive acoustic signals, such as foot steps, sniper shots etc.

The energy-based approach [46, 73], on the other hand, is not susceptible to the problems of time synchronization and on-set detection. It uses the signal strength (i.e., energy) of a received signal to estimate the distance from the target to a sensor. However, the challenges of energy-based localization are two-fold. First, the sensors has to be calibrated in order to accurately estimate the distance. Second, the energy-based localization might suffer from the problem of multi-path propagation in the in-door environment. The application of energy-based localization is more suited for tracking continuous events, such as a moving tank. Both delay-based and energy-based approaches have been implemented on our Mote testbed.

6.2.1 Delay-Based Localization

The acoustic target tracking system built on our testbed consists of multiple static sensory clusters. Each cluster has a cluster head and several slavery acoustic sensors which jointly monitor a specific area. The CH needs to handle a significant amount of computation, which is beyond the capability of the current generation of mica Motes. Therefore, in our implementation, besides a Mote, a PC/104 embedded-PC board [100] equips each CH. The detailed design of the key components for the acoustic tracking system including the time synchronization, onset detection, cross-correlation, and localization, are presented next.

Time Synchronization Accurate timing information is a necessity in the delay-based localization approach. Specifically, all the sensors within the same cluster have to be time-synchronized. The method of time synchronization we adopt is Reference-Broadcast Synchronization (RBS) [21]. In RBS, a CH broadcasts reference radio beacons to its neighbors. Each receiver records the arrival time based on its local clock and sends this information back to the CH. Under the assumption that the broadcasted radio beacon arrives at all receivers simultaneously, after a few rounds of the beacons and replies, the head node can obtain mapping functions of clock readings between any pair of receivers, using statistical methods such as least square linear regression. To this end, the head node can convert any receivers clock readings into a universal clock reading. In our implementation, we have an initialization phase when n ($n = 12$ in our practice) rounds of beacons and replies are performed. After the initialization, timing information can be piggybacked on the pack-



(a) with initialization phase only (b) with calibration phase

Figure 6.1: The problem of clock skew in two motes for the approach with and without calibration phase.

ets exchanged between the cluster head and the sensors. Therefore, the clocks can be calibrated without introducing extra control overheads. The necessity of the calibration phase is shown in the experiment results in Fig. 6.1. The total measurement period shown in x-axis is 1200 seconds. The clock rate used in the system is 2 μs per clock tick. The vertical axis represents the difference in clock readings between two motes. "error" denotes the difference of two clock readings at time t and "avg. abs. error" is the average of the absolute value of the difference in two clock readings in time period $[0,t]$. Fig. 6.1(a) is the result of the approach with only synchronization in the initialization phase but without further calibrations. It clearly shows the problem of clock skew between two Motes. The result of the approach with calibrations is shown in Fig. 6.1 (b). The distortion of clock readings can be kept within 30 μs, which is sufficient for our delay-based acoustic tracking.

Onset Detection Due to the limited computation capability, the current generation of mica motes are not capable of sampling and processing acoustic data concurrently. Instead, major functions such as sensing, wireless transmitting/receiving and processing have to be serialized. Moreover, because of the limited memory in motes, acoustic samples have to be stored in a circular buffer. In order to avoid buffer overflow for useful sample data , an onset detection mechanism is needed to instruct sensors to stop sampling data once the interested acoustic signal is captured. The way a sensor determines whether the incoming acoustic signal is of potential interest is based on the magnitude of the signal. A small sliding window is used to compute the moving average of the magnitude of signals. If the energy within the window exceeds a threshold, the sensor assumes that the current time is close to the onset point of the acoustic signal. The sensor continues recording

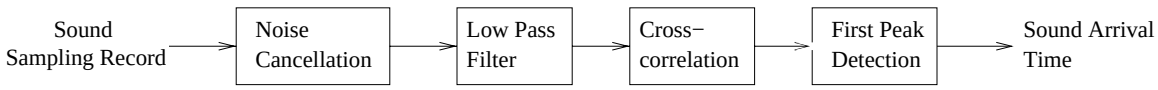


Figure 6.2: Procedures of cross-correlation for sensors

data into the circular buffer until it winds back and reaches a prelude point prior to the onset point. Once the sound of interest is captured, post-processing is conducted separately at both the cluster head and the sensors. The cluster head extracts the sound signature from the recorded samples and broadcasts it to the sensors in its own cluster. Upon receipt of the signature packet from the cluster head, sensors apply cross-correlation to compare the received signature with the buffered data.

Cross-Correlation After receiving the sound signature from the cluster head, each sensor cross-correlates the received signature with buffered data to extract the desired pattern and determine the starting portion of signal. There are several advantages in choosing the starting portion as the reference portion. First, the starting portion is less susceptible to echoes. In in-door environments, the effect of echoes is quite significant. Fortunately, the echo is not presented in the starting portion of acoustic signal unless the sensor is very close to a wall. Second, the uniqueness of onset point and the salient change in sound wave shape at the onset point makes the starting portion very easy to be consistently located among distributed independent sensors. The procedures of cross-correlation are summarized in Fig. 6.2. Two preprocessing procedures are applied to buffered data before cross-correlation. The first step is to remove the interference of noise. The average magnitude of noise is calculated first, and then the samples whose values are close to the average are replaced with the average so that the results of correlation do not ripple due to the oscillation of noises. In the second step, the signal is passed through a second-order Butterworth low-pass filter to remove the high frequency components. After the pre-processing, cross-correlation is applied to the filtered data with the received sound signature to do pattern matching. The final step is to find the first significant peak in the correlation result using a threshold and thus the arrival time is extracted.

Sound Source Localization Upon receiving reply packets including sound arrival time from sensors in the cluster, the CH translates the time into the universal reference time before executing the localization. Localization is done by comparing the differences in sound propagation delays from the source to different acoustic sensors. Suppose the location of sound source is (x, y) and the sound is generated at time t. If a sensor M_1 located at (x_1, y_1) detects the sound at time t_1. We

shall have the following equation:

$$\sqrt{(x-x_1)^2+(y-y_1)^2} = (t_1 - t) \cdot v$$

where v is the velocity of sound. Directly solving the above equation with the readings from three sensors is feasible theoretically but impractical when dealing with the real data since the errors generated in cross-correlation are not negligible. Instead, a robust maximum likelihood (ML) based approach is used in our system. Each pair of arrival time from two sensors determines a hyperbola curve. For examples, two sensor M_1 and M_2, located at (x_1, y_1) and (x_2, y_2), detect the starting points at time t_1 and t_2, respectively. Without loss of generality, suppose that $t_1 > t_2$. The hyperbola curve can be expressed as:

$$\sqrt{(x-x_1)^2+(y-y_1)^2} = \sqrt{(x-x_2)^2+(y-y_2)^2} + (t_1 - t_2) \cdot v$$

Rather than solving a non-linear programming problem, a method of identifying the grid with the most number of hyperbola curves passing through is used to locate the most likely target position. In addition to the localization result, the confidence level of the result can also be interpreted from the above approach and used as an index for data quality in the information-driven routing [82].

Experiment Results The the acoustic localization result within a single cluster is investigated here. Sensors are placed uniformly in a 100×100 $inch^2$ area (see Fig. 6.3). The results of using 8, 10 and 12 sensors per cluster are tested respectively. A PC/104 is placed at the center to serve as the CH. To understand the sensitivity of localization result to the sound source location, 18 locations of sound source shown in Fig. 6.3 are tested. Ten trials are carried out for each of the sound source location. Figure 6.4 gives an example of the tracking results for the case with 12 sensors at 6 out of the total 18 locations. Each "$*$" represents the actual location of the sound source, while each "$\cdot$" corresponds to the localization result in one trial. Fig. 6.5 shows the average sensing error with respect to different sound location and the number of sensors used. In summary, we can locate a sound source with an average error of 13.8 inches, among which 35% of the errors are less than 3 inches and 48.3% of the errors are less than 6 inches.

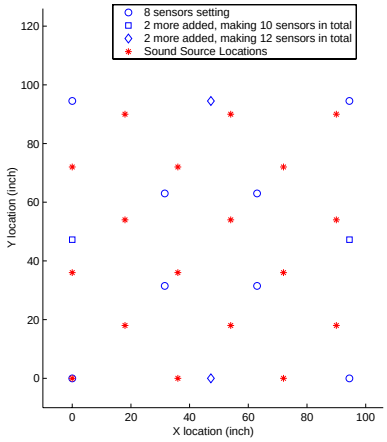


Figure 6.3: Locations of sensors and sound sources in a single cluster.

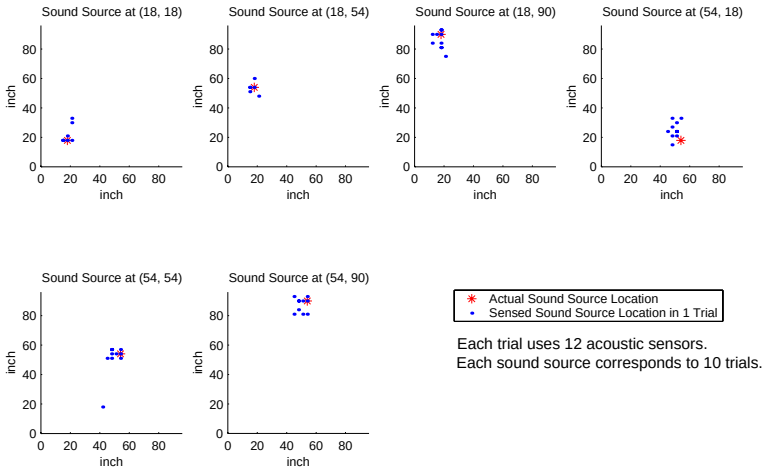


Figure 6.4: An example of triangulation results for different sound source location

6.2.2 Energy-Based Localization

The energy-based localization relies on one physical sound propagation phenomenon: the signal strength (i.e., energy) of a received signal decreases exponentially with the propagation distance shown in Eq. (3.1).

$$r_i = a \cdot \|x - x_i\|^{-\alpha} + n_i, \qquad 1 \leq i \leq N,$$

Therefore, the first step is to test whether the Motes exhibit such a characteristic. In the experiment, twelve sensors are aligned with equal space 2.5 inches. A sound source with a fixed volume is placed from sensors between 2.5 to 30 inches. Fig. 6.6 gives the result of the relationship between the energy readings and the distance ,both in logarithmic scale. Each point is the average result of 40 measurements. Even though this is the result without any calibrations, almost all sensors exhibit

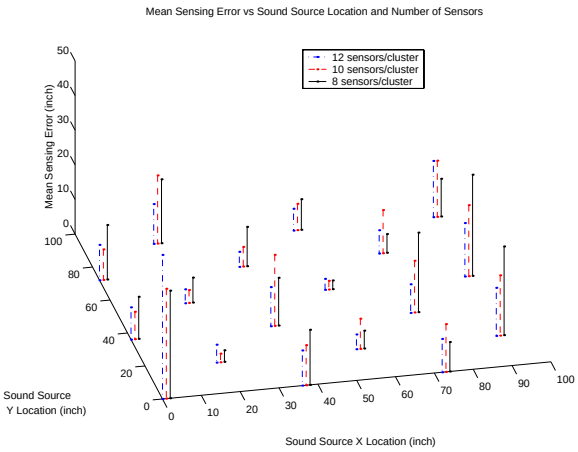


Figure 6.5: Average Error, when Sound Source is at different locations

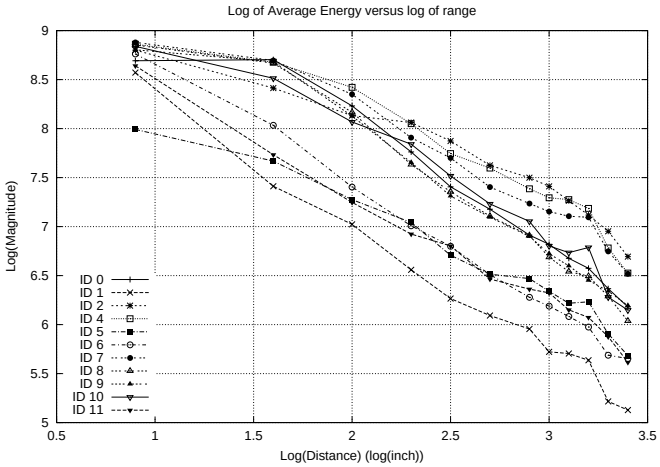


Figure 6.6: The relationship between logarithm of the energy and the logarithm of the distance for 12 motes.

the exponential decay in the relationship between energy and propagation distance. Moreover, we use the least square linear regression method to calculate the attenuation factor, α, approximately equal to 1.0. The next challenge is to calibrate the sensing scale of microphones of different sensors. We take one Mote as a reference Mote and manually adjust the amplifying scale of the microphone of each Mote.

The testbed consists of 18 sensors and 3 CHs deployed in a square grid shown in Fig. 6.7. Each Mote is placed with space 18 inches with neighboring Mote. A simplified version of the dynamic clustering protocol presented in Chapter 3 is implemented. First, the random back-off delay at CH

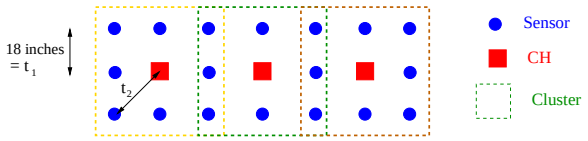


Figure 6.7: System deployment for sensors and CHs in the energy-based localization.

is determined according to Eq. (3.6).

$$D = W_{min} + (W_{max} - W_{min}) \cdot (\frac{d - t_1}{t_2 - t_1}) + U(W_{ran})$$

Second, instead of broadcasting energy and signature information in two phases, only one phase broadcast carrying the energy information is implemented. Upon reception of a request packet, a Mote replies the maximum energy of the window worth of samples stored in the buffer to the active CH. Third, the membership of a sensor is restricted by its location. The whole system is divided into three clusters. Each cluster includes one CH and 8 sensors. A sensor belongs to cluster of its closest CH except for the sensors on the boundary of two clusters which has dynamic membership and can join to the cluster whose CH broadcasts the solicitation packet first. All eight Motes in the active cluster reply the energy packet sequentially (in a TDMA manner). We use Eq. (3.3), a non-linear programming of minimizing the sum of sure of error, to estimate the position of sound sources.

$$(x, y) = \arg\min_{(x,y)} \sum_{i=1}^{m} \frac{(x - o_{x_i})^2 + (y - o_{y_i})^2 - \rho_i^2}{\rho_i^2},$$

The localization errors for sound source at different locations near the central cluster is shown in Fig. 6.8. Each point at a position is the average result of 20 tests. The result shown that when the sound source is placed away from the cluster boundary, the accuracy is very high. On the other hand, if the sound source is at the boundary of the cluster, the error increases dramatically. Especially, the localization error is significant when the "incorrect" cluster is elected. The average localization error for all positions is 8.37 inches, which is satisfactory for the preliminary result.

6.3 Implementation of Utility-Based Data Transport Modules on Motes

In this section, we present an implementation of a simplified version of the data-centric information dissemination mechanism proposed in Chapter 5 on the Motes testbed, including the capacity

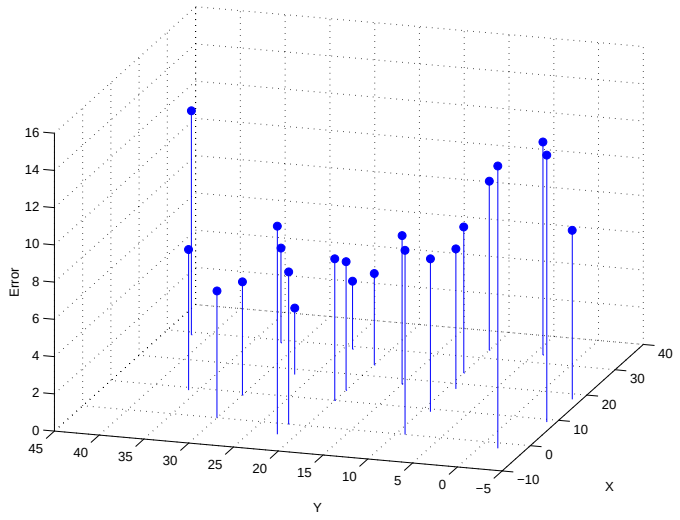


Figure 6.8: Localization result of the case of three clusters when the target is placed near the central cluster.

estimation module and the capacity price controlling module. The dynamic routing algorithm proposed in Chapter 5 is not included in the implementation. A major purpose of the implementation is to propose a generic software architecture with well defined APIs for the sensor network community to realize data-centric information dissemination mechanisms. The software architecture thus laid can be easily adapted to various characteristics and requirements of different applications. An illustrative component of value evaluation and a sample application are provided.

6.3.1 Software Architecture of the Data-Centric Dissemination System

The sample application in our design is a monitoring application that detects the luminosity changes. The Motes are equipped with sensor boards (Mica sensor board) which can collect luminosity information from the environment through photoelectric sensors. The collected sensor readings are transmitted to the sink via the multi-hops communications. Different luminosity readings represent different utility values. By manually changing the luminosity in the vicinity of each sensor, we validate effectiveness of the proposed utility-based data transport protocol.

The components that realize the data-centric dissemination (expressed in the box with oblique lines) reside between the application layer and the routing layer (Fig. 6.9). The component **Surge**, the core component of the application, connecting with **Timer** and **Photo** components periodically receives a luminosity reading from the **Photo** component and sends a packet carrying the reading to the sink via **MultiHopRouter** component. Without the flow control module, the sensing rate

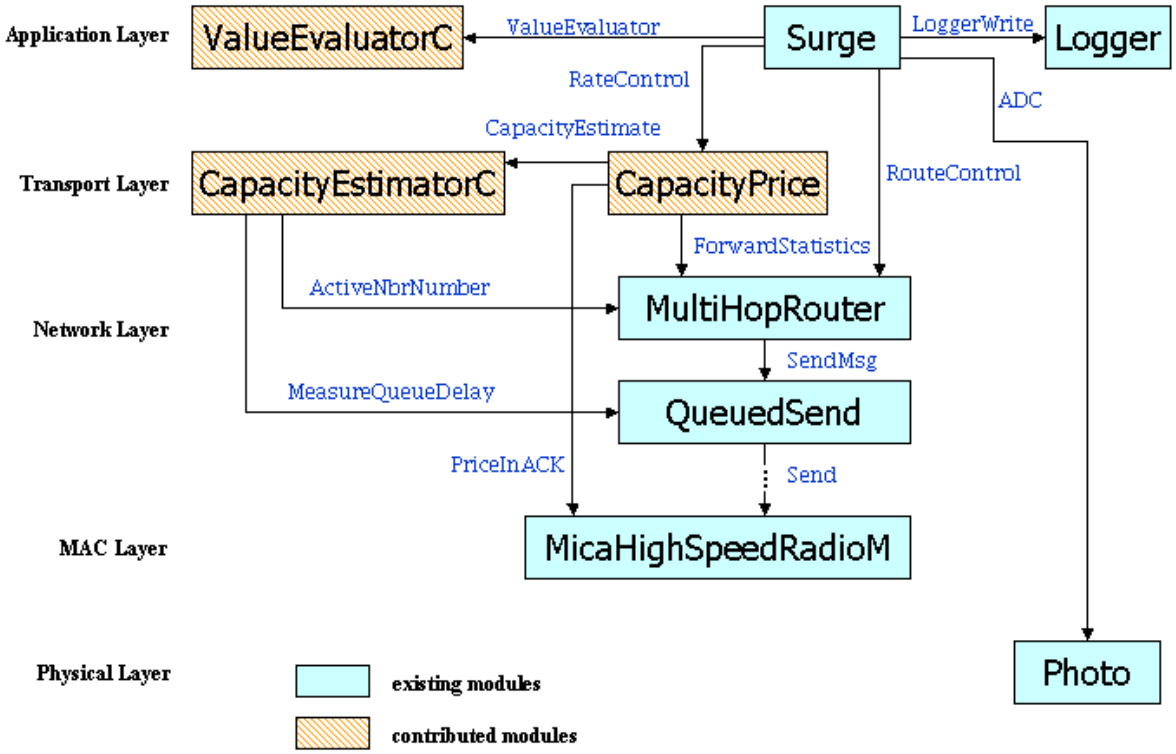


Figure 6.9: The system architecture of the data-centric transport control modules.

is fixed and can be changed only through explicit commands. The proposed quality-centric data dissemination components enable the sensor to determine their own sensing rate based on the value of sensed data and the cost of data dissemination, so as to extract the most amount of information from the sensor network.

The major functions[1]in the created interfaces are listed in the Table 6.1. In the following we introduce the functions and interfaces of each component:

Component ValueEvaluatorC: The function of component **ValueEvaluatorC** is to evaluate the value of the data given by the application. Upon receipt of a reading from **Photo** component, **Surge** calls the command *evaluateValue* in **ValueEvaluatorC**. If the value of the current data is different from the previous one, **ValueEvaluatorC** notifies **Surge** of the value change via the event *notifyValueChange*. The value evaluation method in **ValueEvaluatorC** is devised based

[1]Various kinds of functions specifying the parameter settings are not listed thoroughly.

Interface	Functions
Value Evaluate	command result_t *evaluateValue*(uint16_t data)
	event result_t *notifyValueChange*(uint16_t value)
	command result_t *setLevel*(uint8_t level_num, uint16_t* level, uint16_t* value)
RateControl	command result_t *changeValue*(uint16_t value)
	event result_t *modifyPeriod*(uint16_t new_period)
CapacityEstimate	command result_t *start*()
	event result_t *CapacityChange*(float new_capacity)
ForwardStatistics	command result_t *set_bForwardStatistics*(bool flag)
	event result_t *ForwardRateChange*(uint16_t new_forward_rate)
PriceInACK	command result_t *set_bPriceInACK*(bool flag, uint8_t bit_num, uint8_t my_init_price)
	command result_t *setNewUpPrice*(uint8_t new_up_price)
	event result_t *receivedNewDownPrice*(uint8_t new_down_price)
ActiveNbrNumber	command result_t *set_bActiveNbrnUMBER*(bool flag)
	event result_t *updateActiveNbrNumber*(uint8_t active_nbr_num)
MeasureQueueDelay	command result_t *set_bMeasureQueueDelay*(bool flag)
	event result_t *UpdateThroughput*(float throughput)
	event result_t *BufferOverflow*()
	event result_t *ReportTxFailure*()

Table 6.1: List of main functions in the created interfaces.

on thresholds. The number of levels and their corresponding values can be specified through the command *setLevel*. Since evaluating the value of the data depends on the characteristics of the application, developers can implement their own value evaluation component and replace **ValueEvaluatorC**.

Component CapacityPrice: Component **CapacityPrice** is the core component of the utility-based data transport mechanism. It determines the optimal source rate and the price of the device based on the value of the data and the price of data dissemination. Whenever **Surge** is informed of the value change, it calls the command *changeValue* in **CapacityPrice** to evaluate whether or not the source rate needs to be adjusted. If the result is positive, an event *modifyPeriod* is returned to change the sampling period. Specifically, the changes in the following three aspects could trigger the adjustment of the source rate:

1. Estimation of node capacity: Event *CapacityChange* is sent by component **CapacityEstimatorC** whenever the change in the estimated node capacity exceeds a threshold.

2. Data rate of forwarded flows: Since a sensor could also forward packets on behalf of other sensors to the sink, the change in the data rate of forwarded flows might affect the sensor's price and further affect its optimal source rate. The event is *ForwardRateChange* and is initiated from component **MultiHopRouter**.

3. Price of the relay nodes: Whenever the price from the downstream relay nodes[2] changes, the total price (the sum of the relay price and its own price) is also changed and the optimal source rate has to be adjusted accordingly. Recall in Chapter 5 the price information is piggybacked in the ACK packet in the MAC layer. **CapacityPrice** receives an event *receivedNewDownPrice* from the MAC component **MicaHighSpeedRadioM** when the price of its downstream relay node is changed. In addition, when the sum of the relay price and its own price changes, **CapacityPrice** needs to update its price to be charged to its upstream nodes by calling the function *setNewUpPrice*.

The order of the price and source rate adjustments are different between the former two cases and the latter case. In the first two cases, either the changes in the forwarded rate or the capacity leads to the change of the sensor's price. Therefore, the price adjustment is executed first followed by the source rate adjustment. In the case of the change in the downstream price, the order of adjustments is reversed, i.e. the source rate is first adjusted followed by adjustment of the price. In order to maintain the system stability, both types of adjustments should not be executed too frequently. A maximum update frequency is specified to restrict the number of times updates are executed. On the other hand, if no update has been made within 5 seconds, an event trigger by a timer adjusts the price and the source rate sequentially.

In TinyOS, the MAC component, **MicaHighSpeedRadioM**, uses one byte (0x55) as the ACK packet. In our implementation, 4 (least significant) bits out of the 8 bits are used to carry the price information. That is, the price is encoded at 16 levels. Let P_{min} and P_{max} be the minimum and maximum range of the price. Within the component **CapacityPrice** the price, p is specified as a **float**. On the other hand, the price from the downstream relay nodes received at component **MicaHighSpeedRadioM**, p_{enc}, is specified as an 8-bit integer. The decoding rule is given in Eq. (6.1).

$$p_{dec} = (float)(p_{enc} + 0.5) \cdot \frac{p_{max} - p_{min}}{2^n} + p_{min}, \tag{6.1}$$

where n is the number of bits used to encode the price information. In our implementation, n is equal to 4. Similarly, the price charged to upstream nodes has to be expressed as an 8-bit integer. The encoding rule is given in Eq. (6.2).

$$p_{enc} = (uint8\_t)\frac{p_{dec} - p_{min}}{p_{max} - p_{min}} \cdot 2^n, \tag{6.2}$$

[2]The Downstream relay nodes represent the relay nodes on the path toward the sink.

Component CapacityEstimatorC: The function of component **CapacityEstimatorC** is to report the estimate of the node capacity to the component **CapacityPrice**. Estimate of the node capacity is based on two sources: one is event *updateActiveNbrNumber* from the routing component, **MultiHopRouter**, and the other source is the events from component **QueuedSend**, including *UpdateThroughput*, *BufferOverflow*, and *ReportTxFailure*. The number of active neighbors from the routing component is used to calculate the nominal node capacity, the fair share of the channel capacity constituting the static estimation part of the node capacity. On the other hand, the estimation of the "dynamic" part is based on the information from the component **QueuedSend**. **QueuedSend** provides a buffer at the link layer between the network and MAC layers. It records the times when a packet arrives and when it is sent. With the difference of the two time stamps, the outgoing throughput can be calculated. **QueuedSend** periodically (2.5 seconds) reports the average throughput to **CapacityEstimatorC**. Besides, it reports the erroneous events such as the buffer overflow and the failure to reach the next hop to **CapacityEstimatorC** so that the latter can decrease the estimated node capacity.

Component MultiHopRouter: The routing component used in our implementation is the component **MultiHopRouter**. It is designed for systems with a single sink . The routing module[84] maintains the table of neighbors and the link statistics. The next hop to the destination is the neighbor with the best link quality chosen among the neighbors with the shorter hop count. The link quality is evaluated based on the percentage of successful reception of periodic beacon messages. In our implementation, two additional functions the routing component has to provide with are: (i) the data rate of forwarded flows (ii) the number of active neighbors. Both are evaluated periodically over a period of time . An event is triggered to notify the component **CapacityPrice** and **CapacityEstimatorC** when the the value in the forwarding rate or the number of active neighbors changes, respectively.

Since a node always choose the same node with the best link quality as its forwarder, no flow information has to be kept in the routing table (as was discussed in Chapter 5)

Logger: Io order to obtain the internal state of Motes, it is not sufficient to express the states in 3 LEDs. Instead, important states of debugging information have to be recorded in EEPROM. Each mica Mote has 4kbyte storage capacity in the EEPROM. **Logger** is a component providing the functions to read or write data to the hardware of EEPROM.

6.3.2 Experiment Results

In the experiments, we show the effectiveness of the data-centric utility-based data transport module and compare the performance with a baseline method in which each sensor simply adjusts its sending rate based on the sensor's readings. The the value of sensor data is simply determined based on the thresholds. The relationship between the range of readings and their corresponding values, maximum rates, and minimum rates is given in Table6.2. The luminosity value represents the

Luminosity	Value	Max rate (bps)	Min rate (bps)
> 450	1	50	10
$250 \sim 450$	2	150	10
$100 \sim 250$	5	400	10
< 100	10	800	10

Table 6.2: Threshold based value evaluation and the corresponding maximum and minimum rate.

lightness the sensor detects. The smaller the luminosity value, the darker the environment, and a smaller luminosity value is used to indicate a higher value in our experiment settings. In the experiments, we manually place covers on top of sensor boards to simulate occurrences of events. A thicker cover that blocks most of the light is expected to trigger a higher data rate. We measure and compare the performance of the utility based approach with the baseline approach.

Experiment Testbed Setup The testbed is composed of one sink and multiple Mote sensors. In this system, sensors actively generate data at the rate determined according to the utility of their data and the price they are incurred. The sink, a laptop computer is connected to a Mote (in order to communicate with other Motes) through a serial peripheral interface, and is responsible for collecting data from sensors and display the data on the screen. The User Interface of the system shown in Fig. 6.10 includes a system topology graph and individual node property panels, where each panel gives both the sensor readings and the average rate over the time.

Three types of topologies shown in Fig. 6.11 are tested in the experiments. Fig. 6.11(a) shows the scenario in which all sensors are one-hop away from the sink, while Fig. 6.11(b) and (c) show a line topology and a sink-tree topology, respectively. The routing protocol selects the routes based on the link quality and the underlying routes might be changed over the time, especially in the cases of more complicated interconnection between sensors such as the sink-trees topology in Fig. 6.11(c).

Result in the Star Topology In the case of the star topology where all of the sensors are within one-hop communication range to the sink. The price incurred by a sensor includes only

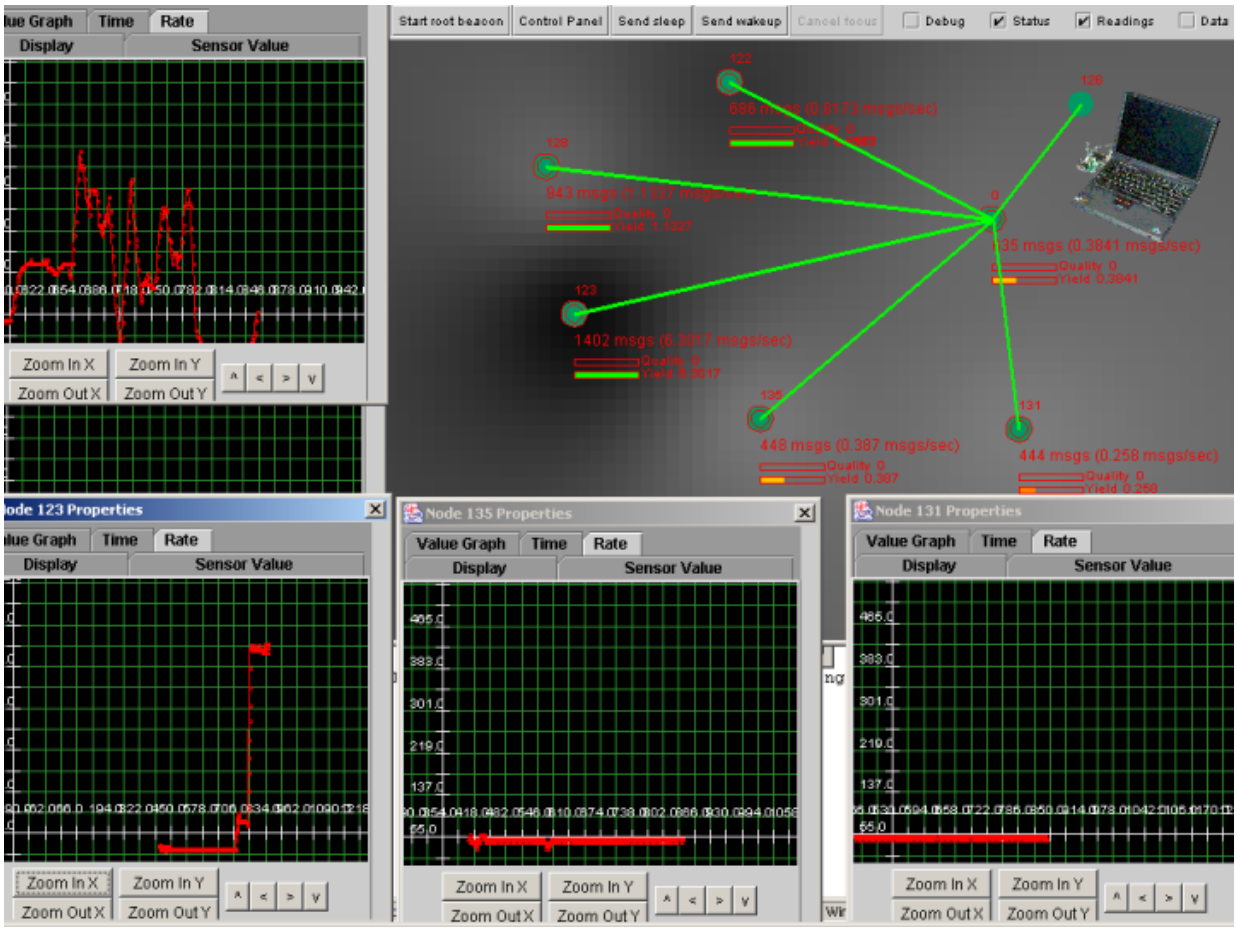
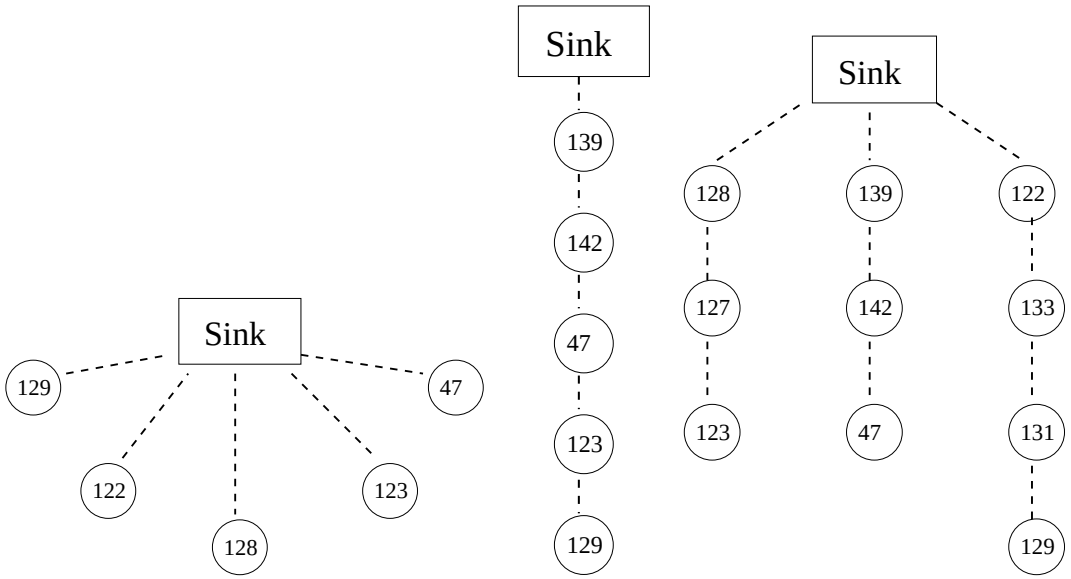


Figure 6.10: The user interface of the data gathering system.

its own price and the price directly from the sink. The way of calculating the capacity price at the sink is different from that at other sensors. Since the sink is only subjected to the connection constraint of serial peripheral interface from the sink Mote to the laptop computer, it uses a fixed capacity constraint (which is set as 4500 bps in the experiments[3]). Other sensors dynamically estimate their effective node capacity via the capacity estimation module. Fig. 6.13 shows the average throughput measured at the sink for all of data flows from 5 one-hop neighbor sensors. Each point in the figure is the average rate measured in a window of 20 seconds (The measurement is performed by further dividing the window into 10 subintervals for rate measurements.) At the beginning no covers are placed on top of any sensor and the data value at each sensor is 1. During the course of measurements, we manually generate five events. At the time of 60 seconds, the first event causes the sensor with ID 128 to generate data with value 2 and the data rate is increased

[3]Even though the capacity of the serial peripheral interface indicated in the specification is 19200 bps, we obtain a effective throughput (excluding overheads) of 4500 bps via measurements. The packet length of each packet is 17 byte.



(a) Star (single-hop) (b) Line (c) Sink-trees

Figure 6.11: Three types of topologies tested in the experiments.

to its maximal allowed rate, 150 bps. Since the total rate of all the flows that are destined for sink is less than its capacity, the minimum price from the sink is charged to all of the sensors. All sensors except sensor 122 send data at their corresponding maximal rate because the flows have not congested the network yet. Sensor 122 exhabits connectivity instability of in the period from $540 - 660$ second.

Result in the Line Topology The average throughput of all the sensors under the utility-based approach in the line topology (Fig. 6.11(b)) is depicted in Fig. 6.12(a). Moreover, the result under the fixed rate approach, in which the maximal allowed rates in Table 6.2 are used for the corresponding value, is presented in Fig. 6.12(b). Under the utility based approach the resource is sufficient to accommodate all of flows before the occurence of the last event. The data rate from sensor with ID 128 decrease dramatically after the last event triggers a data value of 10 at sensor 129. The high flow rate from sensor 129 reduces the flow rates that originate from both sensors 129 and 142. Nevertheless, the rate of sensor 47 is higher than that of sensor 129 in spite of the fact that the data value (=2) of sensor 47 is smaller than the value (=5) of sensor 142. There are two possible reasons to explain this phenomena. First, the operation of increasing the price is performed earlier at sensor 142 than sensor 47 since the overall outgoing rate of sensor 142 is higher than that of sensor 47. Second, the resolution of price information (encoded in 4 bits) is not accurate enough so that the flow rate can not be updated effectively.

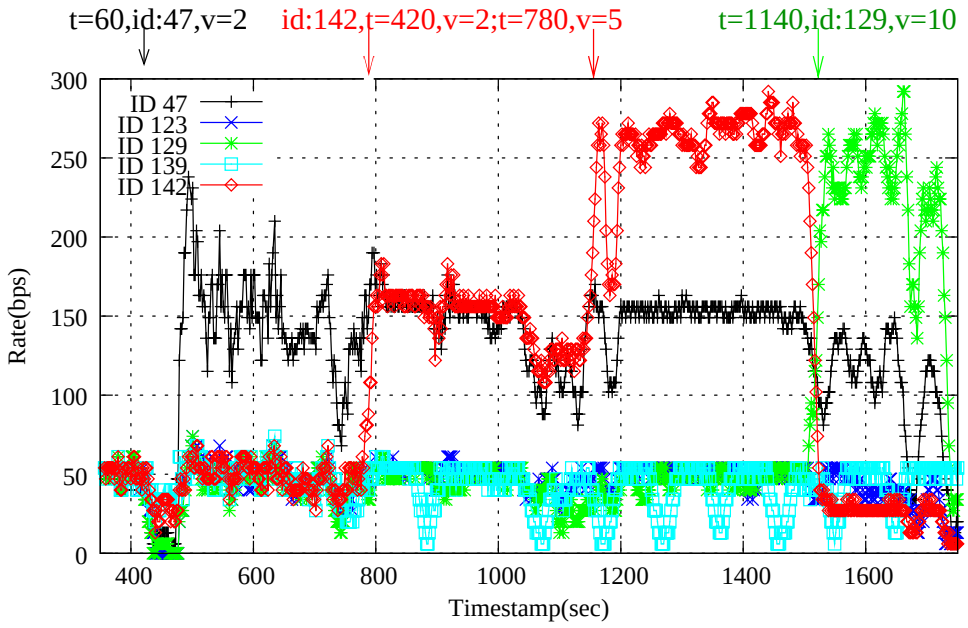
On the other hand, the performance under the fixed rate approach is the same as that under the utility-based approach when there exists plenty of resources. However, when the resource required exceeds the network capacity (the interval after time 780), the fixed rate approach cannot adapt to the abrupt change and can not differentiate data flows with respect to their priority.

Result in the Sink-Trees Topology In the sink-trees topology shown in Fig. 6.11(c), a sensor no longer has a single choice of its next hop, and may be able to reach the sink via different routes. Multiple available routes mitigate the problem of instability of connection. The result under the utility based approach is shown in Fig. 6.14. The rate of flows adapt well to the changes in the data value. Note that at the time 960, the abrupt increase in the rate from sensor 139 congests the network. The data from sensor 127 are buffered [4] but within a short time both flows from sensor 127 and 139 adapt to the changes.

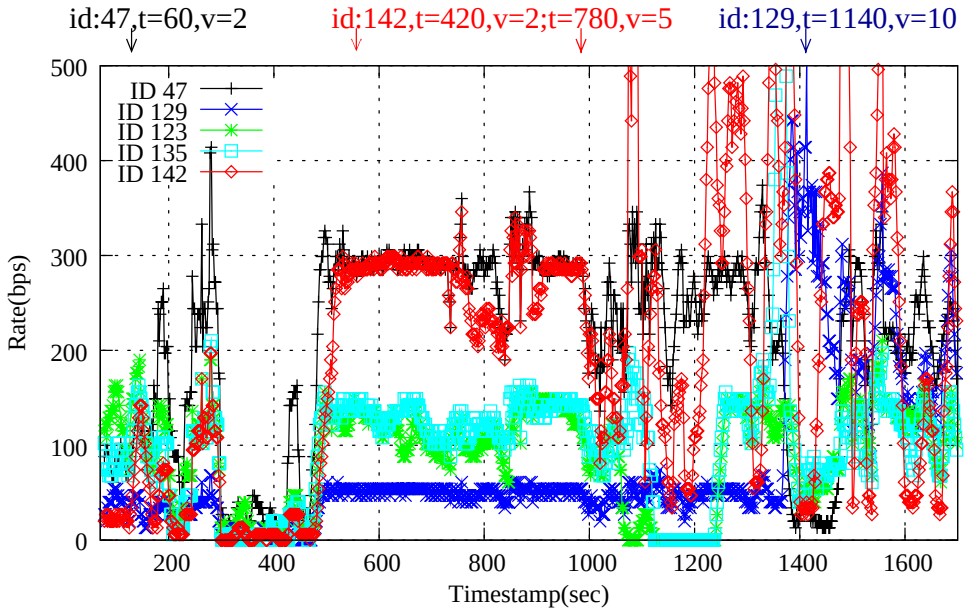
6.4 Summary

In this chapter, we document our the design and implementation of an acoustic tracking system and the utility-based data transport modules on the Motes testbed. In the acoustic tracking system, to achieve high reliability and availability in a system of networked sensors with only limited computation and communication capability, we propose to divide the system into two components, (i) the acoustic target tracking subsystem, and (ii) the communication subsystem. Experimental results using the prototype validate the effectiveness of the proposed design. To facilitate realization of the utility-based data transport modules, we have laid a generic software architecture that comes with a set of well-defined APIs. As demonstrated in the experiments, the proposed software architecture can be easily adapted to various characteristics and requirements of different applications. Also, the utility-based data transport approach does allow flows with data packets with high utility to transmit at higher rates, thus maximizing the total utility.

[4] The buffer size at a node is 16 packets.



(a) utility-based approach



(b) fixed rate approach

Figure 6.12: The average data rate of the line topology for both the utility-based approach and the fixed rate approach.

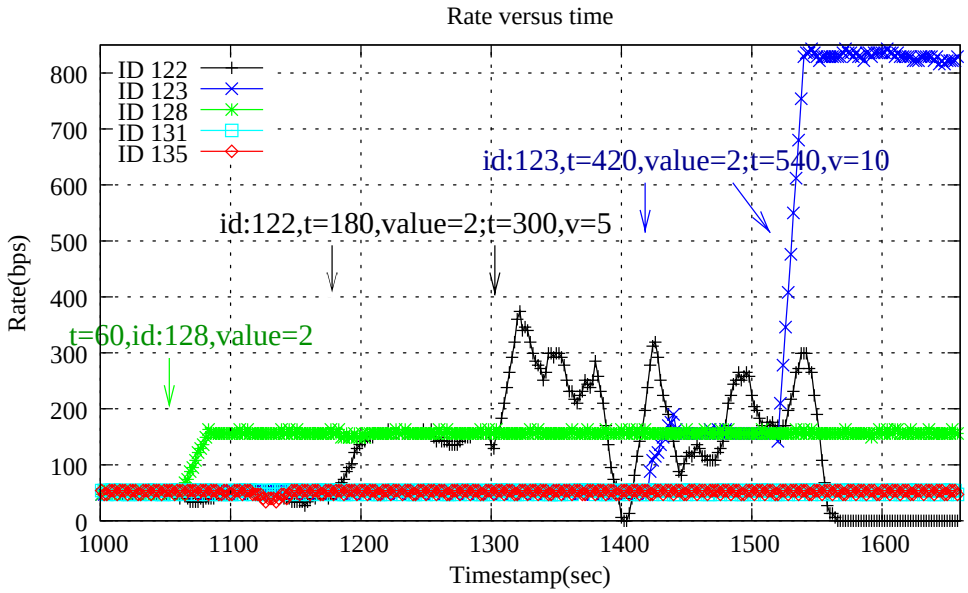


Figure 6.13: The average data rate under the utility-based data transport approach in the star topology.

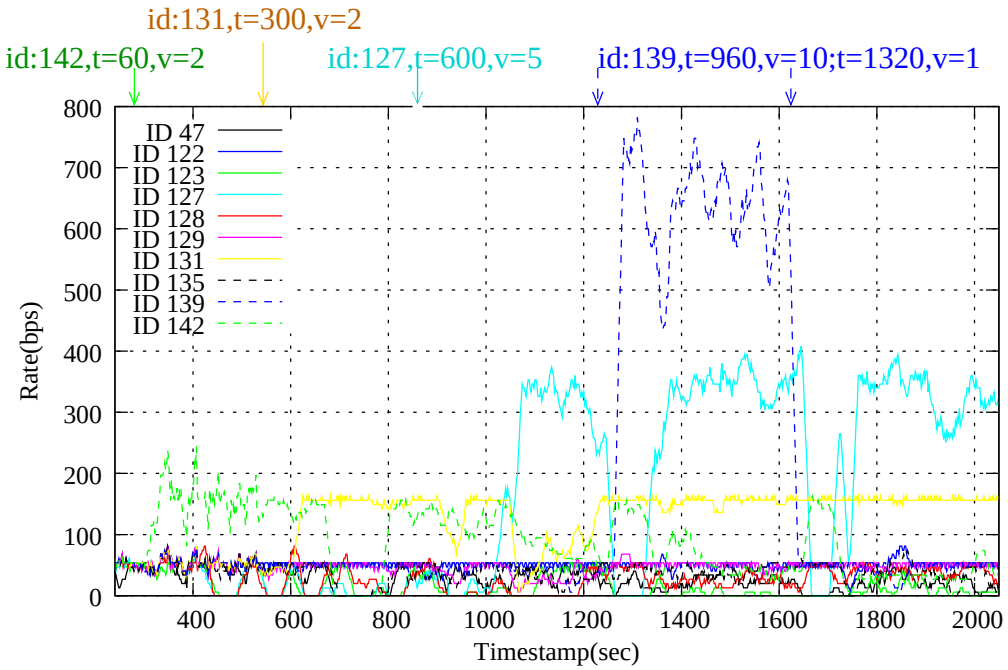


Figure 6.14: The average data rate of the sink-trees topology for the utility-based approach.

Chapter 7

Conclusion and Future Work

In this chapter we first summarize our major contributions of this thesis, and then discuss several research directions for future work.

7.1 Contributions

The main contribution of this thesis is that we propose a comprehensive, data-centric information processing and dissemination framework for sensor networks. In particular, we have:

1. devised a self-organized, dynamic clustering mechanism for target tracking that effectively eliminates contention among sensors and renders more accurate estimates of target locations.

2. formulated the problem of data transport as a utility-based optimization problem, with the objective of maximizing the amount of information (utility) collected at sinks (subscribers), subject to flow conservation, channel bandwidth, and energy constraints. We also devise a distributed algorithm to optimize the utility extraction of the networks.

3. laid a generic software architecture with well-defined APIs for implementing utility-based data transport protocols on Motes testbed. To demonstrate its use, we have implemented and empirically evaluate their performance.

7.2 Future Work

We have identified several avenues for future research. For dynamic clustering the estimated past states can be incorporated in the proposed leader election/sensor reply processes to further reduce the communication overhead and response latency. Second, integrating dynamic clustering with

information quality driven routing protocols is necessary in the presence of multiple targets. Third, how to operate the tracking system in an energy-efficient manner while maintaining high detection rate and low response latency is an important research issue that has not been fully explored.

In the areas of utility-based data-transport, we will continue our research along two thrusts. First, we would like to consider both the flow control and routing problem simultaneously and solve the general formulation in Eqs. (4.1)-(4.4). One serious drawback of the above optimization problem is that the search space is so broad and the control variables, $q_{ij}^{(s)}$, could be of the order of $O(N^3)$ if no constraints are imposed. Several heuristics can facilitate elimination of unlikely solutions and significantly reduce the search space. For instance, based on geographical information, $q_{ij}^{(s)}$ is considered only when nodes i and j are within the radio transmission range and node j is close to one of the sinks than node i is to the source s. With use of this rule, the number of control variables is reduced to $O(N^2)$. Second, the applications considered in the simulation study are monitoring applications in which source that sends the data is static. We will apply the utility based approach to more dynamic applications such as the target tracking system.

Moreover, as mentioned in Section 5.5, the performance of the proposed approach with dynamic routing is not dramatically improved, as the the energy consumed in overhearing is not considered in the energy constraint of the formulated problem. As part of our future work, we plan to incorporate the energy consumed in overhearing into the problem formulation. Alternatively, we may keep the problem formulation unchanged, but instead incorporate power-off operations into the proposed utility-based approach. That is, we turn off the radio circuitry of nodes with low battery levels, if they do not forward packets for other nodes (as determined by the proposed approach). These nodes are then periodically turned on to check if they are on the new routes of certain redirected flows. The interaction between the utility loss due to the latency caused by power-saving operations and the gains in the increased system lifetime is an interesting research issue under investigation.

References

[1] F. Aurenhammer, "Voronoi Diagrams - A Survey Of A Fundamental Geometric Data Structure," *ACM Computing Surveys* 23, pp. 345- 405, 1991.

[2] S.-J. Baek and G. de Veciana and X. Su, "Minimizing energy consumption in large-scale sensor networks through distributed data compression and hierarchical aggregation," IEEE JSAC Special Issue on Fundamental performance limits of wireless sensor networks Vol. 22, No. 6, 1130-1140, August 2004.

[3] S. Bandyopadhyay and E. J. Coyle, "An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks," in IEEE INFOCOM 2003, San Francisco, April 2003.

[4] D. P. Bertsekas, "Nonlinear Programming," 2nd edition, Athena Scientific, Belmont, MA, 1999.

[5] M. Bhardwaj and A.P. Chandrakasan, "Bounding the Lifetime of Sensor Networks Via Optimal Role Assignments," INFOCOM'02, New York, USA, June 2002.

[6] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," in IEEE Journal on Selected Area in Communications Vol.18, No.3, March 2000.

[7] P. Bonnet, J. Gehrke, P. Seshadri, "Querying the Physical World," IEEE Personal Communications Special Issue on Networking the Physical World, 2000.

[8] A. Boulis, S. Ganeriwal, and M.B. Srivastava, "Aggregation in Sensor Networks: An Energy-Accuracy Trade-off," First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA 2003), Anchorage, AK, USA, May 2003.

[9] R. Brooks, C. Griffn, and D. S. Friedlander, "Self-Organized Distributed Sensor Network Entity Tracking," *International Journal of High Performance Computer Applications*, special issue on Sensor Networks, vol. 16, no. 3, Fall 2002.

[10] J. Byers and G. Nasser, "Utility-Based Decision-Making in Wireless Sensor Networks (Extended Abstract)," Proceedings of IEEE MobiHOC 2000, Boston, MA, August 2000.

[11] J.-H. Chang , L. Tassiulas, "Energy Conserving Routing in Wireless Ad-hoc Networks," INFOCOM 2000.

[12] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: an Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," in ACM Wireless Networks Journal, Volume 8, Number 5, September 2002.

[13] W-P Chen, C-J Hou, and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks," in IEEE Transactions on Mobile Computing Special Issue on Sensor Networks, Vol. 3, No. 3, July-September 2004.

[14] W-P Chen and L. Sha, "An Energy-Aware Data-Centric Generic Utility Based Approach in Wireless Sensor Networks," Proc. of the Third International Symposium on Information Processing in Sensor Networks (IPSN), April 2004.

[15] W-P Chen and C-J Hou, L. Sha and M. Caccamo, L. Sha, "A Distributed, Energy-aware, Utility-based Approach for Data Transport in Wireless Sensor Networks," Technical report UIUCDCS-R-2004-2455, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, July 2004.

[16] J. Chou, D. Petrovic, and K. Ramchandran, "A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks," in Proc. IEEE INFOCOM, 2003.

[17] M. Chu, H. Haussecker and F. Zhao, "Scalable Information-driven Sensor Querying and Routing for Ad Hoc Heterogeneous Sensor Networks," *Int'l J. High Performance Computing Applications*, vol. 16, no. 3, Fall 2002.

[18] D. Culler *et al.*, "TinyOS: A Component-Based OS For The Networked Sensor Regime," http://today.cs.berkeley.edu/tos/.

[19] A. Desphande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model Driven Data Acquisition in Sensor Networks," VLDB 2004.

[20] E. Duarte-Melo, M. Liu, and A. Misra, "A computational approach to the joint design of distributed data compression and data dissemination in a data-gathering wireless sensor network," in Proc. 34th Allerton Conf. Communications Control, Monticello, IL, October 2003.

[21] J. Elson, L. Girod, D. Estrin,"Fine-Grained Network Time Synchronization using Reference Broadcasts," in Proc. of the Fifth Symposium on Operating Systems Design and Implementation (OSDI2002), Boston, MA. December 2002.

[22] I. A. Essa, "Ubiquitous Sensing for Smart and Aware Environments," IEEE Personal Communications, October 2000.

[23] D. Estrin, R. Govindan, J. Heidemann and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," In Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCOM '99), August 1999.

[24] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2001.

[25] D. Gay, P. Levis, D. Culler, E. Brewer, "nesC 1.1 language reference manual ," http://webs.cs.berkeley.edu/tos/tinyos-1.x/doc/nesc/ref.pdf, May 2003.

[26] J. Gehrke and S. Madden, "Query Processing In Sensor Networks," Pervasive Computing, 2004.

[27] P. Gupta and P. R. Kumar, The Capacity of Wireless Networks, IEEE Trans. Inform. Theory, 46(2):388-404, 2000.

[28] T. He, J.A. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," Proceedings. of 23rd International Conference on Distributed Computing Systems (ICDCS'03), Providence, Rhode Island, USA. May 2003.

[29] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," Proc. 5th ACM/IEEE MobiCom, Seattle, WA, August 1999.

[30] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocols for Wireless Microsensor Networks," Proc. Hawaaian Int'l Conf. on Systems Science, January 2000.

[31] J. Heidemann, F. Silva, and D. Estrin, "Matching Data Dissemination Algorithms to Application Requirements," The First ACM Conference on Embedded Networked Sensor Systems (Sensys'03), Los Angeles, CA, USA, November 2003.

[32] J. M. Hellerstein, W. Hong, S. Madden, "The sensor spectrum: technology, trends, and requirements," SIGMOD Record 32(4): 22-27, 2003

[33] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, "System Architecture Directions for Networked Sensors," *In Architectural Support for Programming Languages and Operating Systems,* pp. 93-104, 2000.

[34] R. Horst, P.M. Pardalos, and N.V. Thoai, "Introduction to Global Optimization," 2nd ed. Kluwer Academic Publishers, Dordrecht, Boston, London, 2000.

[35] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *In Proc of the Sixth Annual IEEE/ACM International Conference on Mobile Computing and Networking,* August 2000.

[36] S. S. Intille, "Designing a home of the future," IEEE Pervasive Computing, Vol. 1, No. 2 , April-June 2002

[37] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of Interference on Multi-hop Wireless Network Performance,"in ACM MOBICOM, San Diego, CA, September 2003.

[38] K. Kar, S. Sarkar, L. Tassiulas, "Optimization Based Rate Control for Multipath Sessions," Proceedings of Seventeenth International Teletraffic Congress (ITC), Salvador da Bahia, Brazil, December 2001.

[39] B. Karp and H.T. Kung, "Greedy Perimeter Stateless Routing for Wireless Networks," in Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000), Boston, 2000.

[40] F. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," Journal of Operational Research Society, vol. 49, no. 3, March 1998.

[41] H.S. Kim, T.F. Abdelzaher, and W.H. Kwon, "Minimum-Energy Asynchronous Dissemination to Mobile Sinks in Wireless Sensor Networks," The First ACM Conference on Embedded Networked Sensor Systems (Sensys'03), Los Angeles, CA, USA. November 2003.

[42] B. Krishnamachari, D. Estrin, and S. Wicker, "Modelling Data-Centric Routing in Wireless Sensor Networks, INFOCOM'02, New York, USA. June 2002.

[43] P. Krishna, N. H. Vaidya, M. Chatterjee, D. K. Pradhan, "A Cluster-Based Approach for Routing in Dynamic Networks," , ACM SIGCOMM Computer Communications Review, April 1997.

[44] J. Kulik, W. Heinzelman, H. Balakrishnan, "Negotiation-based Protocols for Disseminating Information in Wireless Sensor Networks," Wireless Networks, March 2002.

[45] R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul, and U. Ramachandran, "DFuse: A Framework for Distributed Data Fusion," The First ACM Conference on Embedded Networked Sensor Systems (Sensys'03), Los Angeles, CA, USA. November, 2003.

[46] D. Li, K. Wong, Y. Hu and A. Sayeed, "Detection, Classification, Tracking of Targets in Micro-sensor Networks," *IEEE Signal Processing Magazine*, pp. 17-29, March 2002.

[47] X. Li, Y.J. Kim, R. Govindan and W. Hong, "Multi-dimensional Range Queries in Sensor Networks," Proceedings ACM Conference on Embedded Networked Sensor Systems, November 2003.

[48] R.-F. Liao and A. T. Campbell, "A Utility-Based Approach to Quantitative Adaptation in Wireless Packet Networks," ACM Journal on Wireless Networks (WINET), Vol. 7, No. 5, pp. 541-557, September 2001.

[49] C. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks," IEEE Journal of Selected Areas in Communications, Vol. 15, No.7, September 1997.

[50] S. Lindsey , C. Raghavendra , and K. M. Sivalingam, "Data Gathering Algorithms in Sensor Networks Using Energy Metrics," IEEE Transactions on Parallel and Distributed Systems, v.13 n.9, September 2002.

[51] J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao, "Distributed Group Management for Track Initiation and Maintenance in Target Localization Applications," *Proc. 2nd Workshop on Information Processing in Sensor Networks (IPSN'03)*, April 2003.

[52] S. H. Low and D. E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence," IEEE/ACM Transactions on Networking, 7(6):861-75, December 1999.

[53] H. Luo, P.Medvedev, J. Cheng and S. Lu, "A Self-Coordinating Approach to Distributed Fair Queueing in Ad Hoc Wireless Networks," IEEE INFOCOM 2001, Anchorage, April 2001.

[54] S. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks," 5th Symposium on Operating System Design and Implementation (OSDI 2002), Boston, Massachusetts, USA. USENIX Association. December 2002.

[55] S. Madden, M.J. Franklin, and J.M. Hellerstein, and W. Hong, "The Design of an Acquisitional Query Processor For Sensor Networks," SIGMOD'03, San Diego, CA, June 2003.

[56] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," 2002 ACM International Workshop on Wireless Sensor Networks and Applications September 2002.

[57] J. Mirkovic, G.P. Venkataramani, S. Lu, and L. Zhang, "A Self-Organizing Approach to Data Forwarding in Large-Scale Sensor Networks," IEEE International Conference on Communications (ICC 2001), June 2001.

[58] F. Ordonez and B. Krishnamachari, "Optimal Information Extraction in Energy-Limited Wireless Sensor Networks," IEEE Journal on Selected Areas in Communications, special issue on Fundamental Performance Limits of Wireless Sensor Networks, August 2004.

[59] M.R. Pearlman and Z.J. Haas, "Determining the Optimal Configuration for the Zone Routing Protocol," IEEE Journal on Selected Areas in Communications, Vol. 17, No. 8, August 1999.

[60] C. E. Perkins and E. M. Royer, "Ad hoc On-Demand Distance Vector Routing," Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999.

[61] D. Petrovic, R. C. Shah, K. Ramchandran, and J. Rabaey, "Data Funneling: Routing with Aggregation and Compression for Wireless Sensor Networks," IEEE ICC Int. Workshop on Sensor Network Protocols and Applications, 2003.

[62] Y. Qiu and P. Marbach, "Bandwidth Allocation in Wireless Ad Hoc Networks: A Price-Based Approach," in IEEE INFOCOM 2003, San Francisco, April 2003.

[63] M. Rabbat, and R. Nowak, "Distributed Optimization in Sensor Networks," Proc. of the Third International Symposium on Information Processing in Sensor Networks (IPSN), April 2004.

[64] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-Centric Storage in Sensornets with GHT, a Geographic Hash Table," in Mobile Networks and Applications (MONET), Journal of Special Issues on Mobility of Systems, Users , Data, and Computing: Special Issue on Algorithmic Solutions for Wireless, Mobile, Ad Hoc and Sensor Networks , Kluwer, mid-2003.

[65] N. Sadagopan and B. Krishnamachari, "Maximizing Data Extraction in Energy-Limited Sensor Networks," in IEEE INFOCOM, Hong Kong, March 2004.

[66] Y. Sankarasubramaniam, O.B. Akan, and I.F. Akyildiz, "ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks," Proceedings of ACM MobiHoc'03, Annapolis, Maryland, USA, June 2003.

[67] C. U. Saraydar, N. B. Mandayam, and D. J. Goodman, "Pricing and Power Control in a Multicell Wireless Data Network," IEEE Journal on Selected Areas in Communications, Vol. 19, No. 10, pp. 1883-1892, October 2001.

[68] A. Savvides, C.C. Han and M. B. Srivastava, "Dynamic Fine Grained Localization in Ad-Hoc Sensor Networks," *Proceedings of the Fifth International Conference on Mobile Computing and Networking*, Mobicom 2001, Rome, Italy, July 2001.

[69] A. Scaglione and S. D. Servetto, "On the interdependence of routing and data compression in multi-hop sensor networks," in Proc. ACM Mobicom, Atlanta, GA, 2002.

[70] C. Schurgers , V. Tsiatsis , S. Ganeriwal , and M. Srivastava, "Optimizing Sensor Networks in the Energy-Latency-Density Design Space," IEEE Transactions on Mobile Computing, v.1 n.1, p.70-80, January 2002.

[71] R.C. Shah and J.M. Rabaey, "Energy Aware Routing for Low Energy Ad Hoc Sensor Networks," Wireless Communications and Networking Conference (WCNC2002), IEEE , Volume: 1 , 17-21 March 2002.

[72] C.-C. Shen, C. Srisathapornphat, and C. Jaikeo, "Sensor information networking architecture and applications," IEEE Personal Communications, 8(4):52–59, August 2001.

[73] X. Sheng, and Y-H Hu, "Energy Based Acoustic Source Localization," *Proc. of 2nd Workshop on Information Processing in Sensor Networks (IPSN'03)*, April 2003.

[74] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," In *Proc. of ACM MobiCom'01*, Rome, Italy, July 2001.

[75] D. Slepian and J. K. Wolf, Noiseless encoding of correlated information sources, IEEE Trans. on Inform. Theory, vol. IT-19, pp. 471-480, July 1973.

[76] V. Stanford, "Pervasive computing goes the last hundred feet with RFID systems," IEEE Pervasive Computing, Vol. 2, No. 2, April-June 2003.

[77] S. Tilak, A. Murphy, and W. Heinzelman. Non-uniform Information Dissemination for Sensor Networks. 11th IEEE International Conference on Network Protocols (ICNP'03) , Atlanta, Georgia, USA. November 2003.

[78] R.J. Vanderbei and D.F. Shanno, "An interior-point algorithm for non-convex nonlinear programming," Computational Optimization and Applications, 13:231–252, 1999.

[79] C.Y. Wan, A.T. Campbell, and L. Krishnamurthy, "PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks," WSNA'02, Atlanta, Georgia, USA, September 2002.

[80] C.Y. Wan, S.B. Eisenman, and A.T. Campbell, "CODA: Congestion Detection and Avoidance in Sensor Networks," The First ACM Conference on Embedded Networked Sensor Systems (Sensys03), Los Angeles, CA, USA. November, 2003.

[81] J. Wang, L. Li, S. H. Low and J. C. Doyle, "Can TCP and shortest path routing maximize utility," Proceedings of IEEE INFOCOM, San Francisco, April 2003.

[82] Q. Wang, W-P Chen, R. Zheng, K. Lee, and L. Sha, "Acoustic Target Tracking Using Wireless Sensor Devices," *Proc. of the 2nd Workshop on Information Processing in Sensor Networks (IPSN'03)*, April 2003.

[83] A. Woo and D. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks," Mobicom 2001, Rome, Italy, July 2001.

[84] A. Woo, T. Tong, and D. Culler, "Taming the Underlying Challenges of Reliable Multhop Routing in Sensor Networks," SenSys 2003 Los Angeles, CA, 2003.

[85] Y. Xu, J. Heidemann, D. Estrin, "Geography-informed Energy Conservation for Ad-hoc Routing," *In Proc. of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (ACM MobiCom)*, Rome, Italy, July 2001.

[86] Y. Xue, B. Li, K. Nahrstedt, "Price-based Resource Allocation in Wireless Ad Hoc Networks," in the Proceedings of the Eleventh International Workshop on Quality of Service (IWQoS 2003), 2003.

[87] H. Yang and B. Sikdar, "A Protocol for Tracking Mobile Targets using Sensor Networks," Proceedings of IEEE Workshop on Sensor Network Protocols and Applications (In conjunction with IEEE ICC), Anchogare, AK, May 2003.

[88] Y. Yao and J. Gehrke, "Query Processing for Sensor Networks," First Biennial Conference on Innovative Data Systems Research (CIDR'03). Asilomar, CA, January 2003.

[89] F. Ye, H. Luo, J. Cheng, S. Lu and L. Zhang, "A Two-tier Data Dissemination Model for Large-scale Wireless Sensor Networks," *ACM International Conference on Mobile Computing and Networking (Mobicom 2002)*, Atlanta, Georgia, September 2002.

[90] W. Ye, J. Heidemann, D. Estrin, "An Energy-Efficient MAC protocol for Wireless Sensor Networks,"in *Proc. of INFOCOM'02,* June 2002.

[91] F. Ye, G. Zhong, S. Lu, and L. Zhang, "A Robust Data Delivery Protocol for Large Scale Sensor Networks," 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03), Palo Alto, CA, April 2003.

[92] Y. Yu, B. Krishnamachari, and V.K. Prasanna, "Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks," IEEE Infocom'04, Hong Kong, March 2004.

[93] W. Zhang, and G. Cao, "DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks," to appear in IEEE Transactions on Wireless Communication.

[94] F. Zhao, J. Shin and J. Reich, "Information-Driven Dynamic Sensor Collaboration for Tracking Applications," *IEEE Signal Processing Magazine*, March 2002.

[95] Crossbow Technology, Inc. http://www.xbow.com/support/support_pdf_files/mts-mda_series_user_manual_revb.pdf.

[96] Infrarad Sensor. http://www.interq.or.jp/japan/se-inoue/e_pyro.htm.

[97] KEYENCE America. http://www.keyence.com/products/sensors.html.

[98] UCB, LBNL, "VINT network simulator," http://www-mash.cs.berkeley.edu/ns/.

[99] UCLA, "SensorSim : A Simulation Framework for Sensor Networks," http://nesl.ee.ucla.edu/projects/sensorsim/.

[100] PC/104 Embedded-PC Modules http://www.pc104.org/.

[101] UCB, "TinyDB," http://telegraph.cs.berkeley.edu/tinydb/index.htm.

[102] UCB, "TinyOS," http://www.tinyos.net.

Vita

Wei-Peng Chen was born in Tainan city, Taiwan. He received the BS degree in electrical engineering from National Taiwan University in 1994 and the MS degree in electrical engineering from National Taiwan University and the Ohio State University, in 1996 and 2001. He joined Department of Computer Science at the University of Illinois at Urbana-Champaign in 2001. From 1998-1999, he worked at Philips Innovation Center at Taipei as a research engineer. His research interests include mobile and ad hoc networks, sensor networks, seamless and location-aware computing, and internet applications.